

A. Preprocessing

Data diperoleh dari Badan Pusat Statistik



BADAN PUSAT

STATISTIK

Beranda

Rencana Terbit

Produk ▾

Layanan

Informasi Publik ▾









Statistik Demografi dan Sosial

▾



Statistik Ekonomi

▴

Statistik Makroekonomi

Neraca Ekonomi

Statistik Bisnis

Statistik sektoral

Keuangan Pemerintah, Fiskal dan Statistik Sektor Publik

Perdagangan Internasional dan Neraca Pembayaran

Jumlah Penumpang Pesawat di Bandara Utama (Orang), 2024

Terakhir Diperbarui : 2 Desember 2024

← Kembali

Unduh ▾

</> JSON

↻ Bagikan

2024 ▾

Cari data statistik

☐

Freeze judul kolom

Bandara Utama Domestik	Keberangkatan pada Penerbangan Domestik									
	2024									
	Januari	Februari	Maret	April	Mei	Juni	Juli	Agustus	September	Oktober
	10,000	12,000	15,000	18,000	20,000	22,000	25,000	28,000	30,000	32,000

Gambar 1 Sumber data

Kami mengumpulkan data 5 tahun terakhir untuk penelitian kali ini, yaitu data tahun 2020 hingga 2024, diman data tersedai dari bulan Januari 2020 hingga Oktober 2024.

[illegible]

Gambar 2 Contoh data mentah

Kode program:

```
eksport_to_csv.py > ...
1 import pandas as pd
2 from glob import glob
3
4 file_paths = glob("data_*.csv")
5 data_internasional = []
6 data_domestik = []
7
```

Gambar 3 Membaca data dari banyak file

```

10 for file in file_paths:
11     df = pd.read_csv(file, skiprows=4)
12     df.rename(columns={"Unnamed: 0": "Bandara"}, inplace=True)
13
14     # Ambil tahun dari nama file
15     tahun = file.split("_")[1].split(".")[0]
16
17     # Hapus kolom yang tidak dibutuhkan
18     df = df[~df["Bandara"].isin(["TOTAL", "Bandara Lainnya"])]
19
20     # Atur kolom domestik dan internasional
21     domestik_cols = ["Bandara"] + bulan_list
22     internasional_cols = ["Bandara"] + [f"{bulan}.1" for bulan in bulan_list]
23
24     # Proses data domestik
25     df_domestik = df[domestik_cols].copy()
26     df_domestik = df_domestik.melt(id_vars=["Bandara"], var_name="Bulan", value_name="Jumlah")
27     df_domestik["Jenis"] = "Domestik"
28     df_domestik["Tahun"] = tahun
29     data_domestik.append(df_domestik)
30
31     # Proses data internasional
32     df_internasional = df[["Bandara"] + internasional_cols[1:]].copy()
33     df_internasional.columns = domestik_cols # Samakan header dengan domestik
34     df_internasional = df_internasional.melt(id_vars=["Bandara"], var_name="Bulan", value_name="Jumlah")
35     df_internasional["Jenis"] = "Internasional"
36     df_internasional["Tahun"] = tahun
37     data_internasional.append(df_internasional)

```

Gambar 4 Merapikan format data

```

39 # Gabungkan semua data
40 df_internasional_all = pd.concat(data_internasional, ignore_index=True)
41 df_domestik_all = pd.concat(data_domestik, ignore_index=True)
42 df_all = pd.concat([df_internasional_all, df_domestik_all], ignore_index=True)
43
44 # Pivot data untuk format akhir
45 df_pivot_all = df_all.pivot(index=["Tahun", "Bulan", "Jenis"], columns="Bandara", values="Jumlah").reset_index()

```

Gambar 5 Penggabungan data

```

47 # Penanganan missing value
48 df_pivot_all.replace("-", pd.NA, inplace=True)
49 df_pivot_all = df_pivot_all.loc[:, ~df_pivot_all.columns.isna()]
50
51 df_pivot_all = df_pivot_all.dropna(subset=["Hasanudin-Makassar", "Juanda-Surabaya", "Kualanamu-Medan", "Ngurah Rai-Bali", "Soekarno Hatta-Jakarta"])
52
53 df_pivot_all["Hasanudin-Makassar"] = df_pivot_all["Hasanudin-Makassar"].fillna(0)

```

Gambar 6 Penanganan missing values

	Tahun	Bulan	Jenis	Hasanudin-Makassar	Juanda-Surabaya	Kualanamu-Medan	Ngurah Rai-Bali	Soekarno Hatta-Jakarta
1	2020	Agustus	Domestik	134043	186467	89451	83260	574597
2	2020	Agustus	Internasional	0	293	222	1461	28513
3	2020	April	Domestik	49046	97748	29386	44122	191002
4	2020	April	Internasional	0	1026	616	6752	17499
5	2020	Desember	Domestik	194363	244579	133574	189289	931481
6	2020	Desember	Internasional	0	826	510	196	56330
7	2020	Februari	Domestik	273885	481881	227602	346962	1551967
8	2020	Februari	Internasional	0	71723	63610	425633	444267
9	2020	Januari	Domestik	308503	553747	288819	453130	1600594
10	2020	Januari	Internasional	0	98475	96538	641039	626097

Gambar 7 Hasil akhir

Data hasil preprocessing berjumlah 124 baris data, dengan fitur berupa tahun, bulan, jenis, lalu nama-nama bandara (Hasanudin-Makassar, Juanda-Surabaya, Kualanamu-Medan, Ngurah Rai-Bali, dan Soekarno Hatta-Jakarta) yang masing masing berisi jumlah penumpang (orang) yang berangkat dari sana.

Pada preprocessing, data mentah seperti pada gambar (2) disusun dengan baris kode dari gambar (4), kemudian digabungkan oleh abris kode pada gambar (5). Disini ditemukan missing values, diantaranya data penerbangan pada bandara Hasanuddin-Makassar pada tahun 2020 yang tidak ada. Missing value ini ditanggulangi dengan memberikan nilai 0 karena memang pada waktu tersebut 0 orang yang berangkat (belum dibuka). Kemudian Missing value berikutnya ada pada penerbangan November dan Desember 2024 yang belum tersedia. Missing value ini ditanggulangi dengan menghapus baris tersebut dari data. Hasilnya data siap untuk dipakai, gambar (7).

B. Feature Engineering

Setelah data disimpan dalam sebuah file yang rapi, data masih perlu diolah sedikit lagi untuk digunakan dalam melatih model. Pada tahap ini, data dikonversi menjadi data numerik, dan terdapat penambahan fitur baru, yaitu lag. Lag adalah kumpulan data sebelumnya. Misalnya, jika kita ingin memprediksi penumpang pada Februari 2024, Data lag terdiri dari data penumpang Januari 2024, Desember 2023, dan seterusnya.

```
15 # Load data
16 df = pd.read_csv('../datafinal.csv')
17
18 # Konversi kolom menjadi numerik
19 bulan_mapping = {
20     "Januari": 1, "Februari": 2, "Maret": 3, "April": 4, "Mei": 5, "Juni": 6,
21     "Juli": 7, "Agustus": 8, "September": 9, "Oktober": 10, "November": 11, "Desember": 12
22 }
23 df["Bulan"] = df["Bulan"].map(bulan_mapping)
24
25 df["Jenis"] = df["Jenis"].map({"Domestik": 1, "Internasional": 0})
26 # Pastikan data sudah numerik
27 df.iloc[:, 3:] = df.iloc[:, 3:].apply(pd.to_numeric, errors='coerce')
28
```

Gambar 8 Konversi ke data numerik

```
28 # Siapkan fitur
29 bandara_list = df.columns[3:] # Daftar semua bandara
30 for bandara in bandara_list:
31     df[f"{bandara}_lag"] = df[bandara].shift(1).fillna(0)
32
33 # Pilih fitur untuk semua bandara
34 x = df[["Tahun", "Bulan", "Jenis"] + [f"{bandara}_lag" for bandara in bandara_list]]
35 y = df[bandara_list[:5]] # Menggunakan target untuk semua bandara
36
37 # Split data menjadi training dan testing set
38 scaler = MinMaxScaler()
39 x_scaled = scaler.fit_transform(x)
40 x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
```

Gambar 9 Feature engineering

```
Nama field untuk fitur (features):  
['Tahun', 'Bulan', 'Jenis', 'Hasanudin-Makassar_lag', 'Juanda-Surabaya_lag', 'Kualanamu-Medan_lag', 'Ngurah Rai-Bali_lag',  
 'Soekarno Hatta-Jakarta_lag']  
Nama field untuk target:  
['Hasanudin-Makassar', 'Juanda-Surabaya', 'Kualanamu-Medan', 'Ngurah Rai-Bali', 'Soekarno Hatta-Jakarta']
```

Gambar 10 Fitur dan target

C. Pemilihan Model

Disini kami memilih beberapa model dan menggunakan beberapa metrik untuk melihat hasil evaluasinya. Metrik yang digunakan yaitu:

1. Mean Absolute Error (MAE)

- **Definisi:** Rata-rata dari nilai absolut perbedaan antara nilai prediksi dan nilai aktual.
- **Interpretasi:** Mengukur seberapa besar kesalahan prediksi rata-rata tanpa memperhatikan arah kesalahan (positif atau negatif).
- **Nilai:** Semakin kecil MAE, semakin baik model Anda.

2. Mean Squared Error (MSE)

- **Definisi:** Rata-rata dari kuadrat perbedaan antara nilai prediksi dan nilai aktual.
- **Interpretasi:** Memberikan penalti lebih besar pada kesalahan yang lebih besar karena menggunakan kuadrat. Cocok untuk mendeteksi outlier.
- **Nilai:** Semakin kecil MSE, semakin baik.

3. Root Mean Squared Error (RMSE)

- **Definisi:** Akar kuadrat dari MSE.
- **Interpretasi:** Sama seperti MSE tetapi dalam satuan yang sama dengan data asli, sehingga lebih mudah diinterpretasikan.
- **Nilai:** Semakin kecil RMSE, semakin baik.

4. R² Score (R-Squared)

- **Definisi:** Proporsi variabilitas dalam data yang dapat dijelaskan oleh model.
- **Range:** Dari $-\infty$ hingga 1. Nilai 1 berarti model sempurna, sedangkan nilai 0 berarti model tidak lebih baik daripada rata-rata.
- **Interpretasi:** Seberapa baik model Anda menjelaskan data dibandingkan dengan model sederhana (rata-rata).

5. Explained Variance Score

- **Definisi:** Mengukur proporsi varians yang dijelaskan oleh model prediksi.
- **Range:** 0 hingga 1. Semakin mendekati 1, semakin baik model dalam menjelaskan data.

- **Interpretasi:** Mirip dengan R^2 , tetapi lebih toleran terhadap prediksi buruk di data tertentu.

6. Median Absolute Error

- **Definisi:** Median dari nilai absolut perbedaan antara nilai prediksi dan nilai aktual.
- **Interpretasi:** Sama seperti MAE, tetapi menggunakan median sehingga lebih tahan terhadap outlier.
- **Nilai:** Semakin kecil, semakin baik.

7. Akurasi untuk Toleransi $\pm 5\%$, $\pm 10\%$, $\pm 15\%$

- **Definisi:** Persentase prediksi yang berada dalam rentang toleransi tertentu dari nilai aktual.
- **Interpretasi:** Misalnya, untuk toleransi $\pm 5\%$, berapa persen prediksi yang memiliki selisih kurang dari 5% dari nilai aktual.
- **Nilai:** Semakin tinggi, semakin baik.

Hasil evaluasi

```
C:\Kuliah\Semester5\MachineLearning\Project Akhir\linear_reggress>python modeling.py
Mean Absolute Error (MAE): 434126.7424375618
Mean Squared Error (MSE): 2031359759687.7356
Root Mean Squared Error (RMSE): 1425257.7870994902
R^2 Score: 0.1216905070004098
Explained Variance Score: 0.12271918372254338
Median Absolute Error: 209105.0540248013
Akurasi untuk toleransi  $\pm 5.0\%$ : 0.80%
Akurasi untuk toleransi  $\pm 10.0\%$ : 1.60%
Akurasi untuk toleransi  $\pm 15.0\%$ : 5.60%
```

Gambar 11 Evaluasi Regresi Linear

```
C:\Kuliah\Semester5\MachineLearning\Project Akhir\regresion>python modeling.py
Mean Absolute Error (MAE): 150871.11983999997
Mean Squared Error (MSE): 206770371230.6543
Root Mean Squared Error (RMSE): 454720.10207451164
R^2 Score: 0.903542578852184
Explained Variance Score: 0.9039390409549757
Median Absolute Error: 44897.768
Akurasi untuk toleransi  $\pm 5.0\%$ : 16.00%
Akurasi untuk toleransi  $\pm 10.0\%$ : 30.40%
Akurasi untuk toleransi  $\pm 15.0\%$ : 34.40%
```

Gambar 12 Evaluasi Random Forest

```

C:\Kuliah\Semester5\MachineLearning\Project Akhir\random_forest>python modeling.py
Mean Absolute Error (MAE): 568909.11904
Mean Squared Error (MSE): 2711192022835.544
Root Mean Squared Error (RMSE): 1646569.774663541
R^2 Score: -0.17869595599463084
Explained Variance Score: -0.15486645832169574
Median Absolute Error: 81910.44200000001
Akurasi untuk toleransi ±5.0%: 11.20%
Akurasi untuk toleransi ±10.0%: 22.40%
Akurasi untuk toleransi ±15.0%: 25.60%

```

Gambar 13 Evaluasi Random Forest + Imputer

```

C:\Kuliah\Semester5\MachineLearning\Project Akhir\XGBRegressor>python modeling.py
Mean Absolute Error (MAE): 729258.7881948624
Mean Squared Error (MSE): 6438935848309.428
Root Mean Squared Error (RMSE): 2537505.8321724953
R^2 Score: -1.3962130546569824
Explained Variance Score: -1.3593291621550019
Median Absolute Error: 36493.134375
Akurasi untuk toleransi ±5.0%: 13.60%
Akurasi untuk toleransi ±10.0%: 24.00%
Akurasi untuk toleransi ±15.0%: 30.40%

```

Gambar 14 Evaluasi XGB Regressor

Berdasarkan gambar (11) sampai gambar (14), model yang memberikan hasil paling memuaskan adalah model random forest. Namun, penggunaan imputer (untuk mengubah nilai yang kosong menjadi berisi nilai rata-rata) malah menurunkan performanya. Di sisi lain, XGB Regressor memiliki akurasi yang lumayan dibanding model sebelumnya, namun R^2 Score bernilai negatif, dimana berarti model gagal menjelaskan data.

Penyesuaian model Random Forest

```

from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

```

Gambar 15 Random forest

Random forest adalah kumpulan decision tree. N estimator dalam hal ini adalah jumlah decision tree yang dibuat. Secara default n estimator bernilai 100. Namun, kita bisa meningkatkan jumlah decision tree dalam random forest.

```
Mean Absolute Error (MAE): 150871.11983999997
Mean Squared Error (MSE): 206770371230.6543
Root Mean Squared Error (RMSE): 454720.10207451164
R^2 Score: 0.903542578852184
Explained Variance Score: 0.9039390409549757
Median Absolute Error: 44897.768
Akurasi untuk toleransi  $\pm 5.0\%$ : 16.00%
Akurasi untuk toleransi  $\pm 10.0\%$ : 30.40%
Akurasi untuk toleransi  $\pm 15.0\%$ : 34.40%
Akurasi untuk toleransi  $\pm 20\%$ : 37.60%
```

Gambar 16 Random Forest dengan 100 decision tree

```
Mean Absolute Error (MAE): 159506.13412
Mean Squared Error (MSE): 231465757506.46176
Root Mean Squared Error (RMSE): 481108.88321300177
R^2 Score: 0.889986043620776
Explained Variance Score: 0.890466530263164
Median Absolute Error: 44076.969999999999
Akurasi untuk toleransi  $\pm 5.0\%$ : 16.80%
Akurasi untuk toleransi  $\pm 10.0\%$ : 31.20%
Akurasi untuk toleransi  $\pm 15.0\%$ : 35.20%
Akurasi untuk toleransi  $\pm 20\%$ : 38.40%
```

Gambar 17 Random Forest dengan 200 decision tree

```
Mean Absolute Error (MAE): 158538.8813066667
Mean Squared Error (MSE): 230063466934.8077
Root Mean Squared Error (RMSE): 479649.31662080757
R^2 Score: 0.8899051049537521
Explained Variance Score: 0.8903905866690304
Median Absolute Error: 41974.301333333344
Akurasi untuk toleransi  $\pm 5.0\%$ : 16.00%
Akurasi untuk toleransi  $\pm 10.0\%$ : 32.00%
Akurasi untuk toleransi  $\pm 15.0\%$ : 35.20%
Akurasi untuk toleransi  $\pm 20\%$ : 40.80%
```

Gambar 18 Random Forest dengan 300 decision tree


```
Mean Absolute Error (MAE): 156692.89899999998
Mean Squared Error (MSE): 213149592521.25308
Root Mean Squared Error (RMSE): 461681.26724099723
R^2 Score: 0.8939617832202889
Explained Variance Score: 0.8945576678639544
Median Absolute Error: 43594.39100000002
Akurasi untuk toleransi  $\pm 5.0\%$ : 14.40%
Akurasi untuk toleransi  $\pm 10.0\%$ : 30.40%
Akurasi untuk toleransi  $\pm 15.0\%$ : 35.20%
Akurasi untuk toleransi  $\pm 20\%$ : 40.80%
```

Gambar 19 Random Forest dengan 400 decision tree

```
Mean Absolute Error (MAE): 155879.76136
Mean Squared Error (MSE): 214237070773.58377
Root Mean Squared Error (RMSE): 462857.5059060658
R^2 Score: 0.8963435143283233
Explained Variance Score: 0.8968331090938871
Median Absolute Error: 45047.118399999999
Akurasi untuk toleransi  $\pm 5.0\%$ : 12.80%
Akurasi untuk toleransi  $\pm 10.0\%$ : 28.80%
Akurasi untuk toleransi  $\pm 15.0\%$ : 36.00%
Akurasi untuk toleransi  $\pm 20\%$ : 40.80%
```

Gambar 20 Random Forest dengan 500 decision tree

Berdasarkan hasil evaluasi, dapat dilihat MAE malah meningkat dari 100 decision tree ke 200 decision tree, dan perlahan menurun di 300 hingga 500. Namun, di sisi lain, Median Absolute Error menurun dan akurasi meningkat dari 100 decision tree ke 200 decision tree. Dari segi akurasi untuk toleransi rendah, 200 decision tree memiliki nilai tertinggi, dan menambah jumlah decision tree justru menurunkan nilainya kembali. Untuk toleransi lebih renggang (20%), nilai akurasi nampak tidak jauh berubah dari 38,4% ke 40,8%, dan konsisten di 40,8% meskipun jumlah decision tree ditingkatkan.

Peningkatan performa:

```
# Normalisasi data
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Split data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Gambar 21 Scaling


```
Mean Absolute Error (MAE): 156328.1314133333
Mean Squared Error (MSE): 218517632517.16977
Root Mean Squared Error (RMSE): 467458.69605471
R^2 Score: 0.8983049382927206
Explained Variance Score: 0.8987103399533269
Median Absolute Error: 40942.475999999995
Akurasi untuk toleransi ±5.0%: 17.60%
Akurasi untuk toleransi ±10.0%: 28.00%
Akurasi untuk toleransi ±15.0%: 36.00%
Akurasi untuk toleransi ±20.0%: 42.40%
```

Gambar 22 Evaluasi setelah discaling

Berdasarkan gambar (22), dapat dilihat akurasi yang sebelumnya pada gambar (17), yang menggunakan 200 decision tree, mengalami peningkatan penurunan error secara keseluruhan, dan mengalami peningkatan R^2 score dan peningkatan akurasi.

D. Menyimpan Model dan Scaler

Untuk mengimplementasikan model yang sudah ada, model dan scaler disimpan dalam file .pkl. File .pkl adalah file yang digunakan untuk menyimpan objek Python dalam format yang telah "dipickling" menggunakan modul pickle. Pickling adalah proses serialisasi objek Python, sehingga dapat disimpan ke dalam file dan dimuat kembali untuk digunakan di lain waktu.

```
import pickle

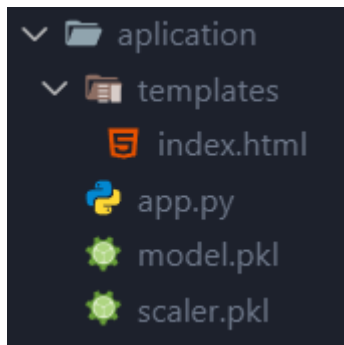
# Menyimpan model ke dalam file
with open('model.pkl', 'wb') as f:
    pickle.dump(rf_model, f)

# Menyimpan scaler yang digunakan untuk normalisasi
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)
```

Gambar 23 Menyimpan model dan scaler

E. Implementasi

Model yang sudah disimpan akan digunakan dalam aplikasi berbasis web dengan framework flask. Struktur folder disiapkan seperti gambar (24). Model dibaca gambar (25). Endpoint dibuat sederhana saja, dengan default HTTP method, get, yang menampilkan form dari template index.html, lalu jika form disubmit, http menjadi post, dan input diproses untuk menghasilkan output di halaman yang sama, seperti kode pada gambar (26) dan tampilannya pada gambar (27) dan gambar (28)



Gambar 24 Struktur folder implementasi

```

1  from flask import Flask, render_template, request
2  import pickle
3
4  app = Flask(__name__)
5
6  # Load model dan scaler
7  with open('model.pkl', 'rb') as model_file:
8      model = pickle.load(model_file)
9
10 with open('scaler.pkl', 'rb') as scaler_file:
11     scaler = pickle.load(scaler_file)

```

Gambar 25 Model dan scaler diupload di aplikasi

```

@app.route('/', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        tahun = int(request.form['tahun'])
        bulan = int(request.form['bulan'])
        jenis = int(request.form['jenis'])
        target = request.form['target']

        features = [tahun, bulan, jenis]
        for t in targets:
            features.append(0 if t != target else 1)

        features_scaled = scaler.transform([features])

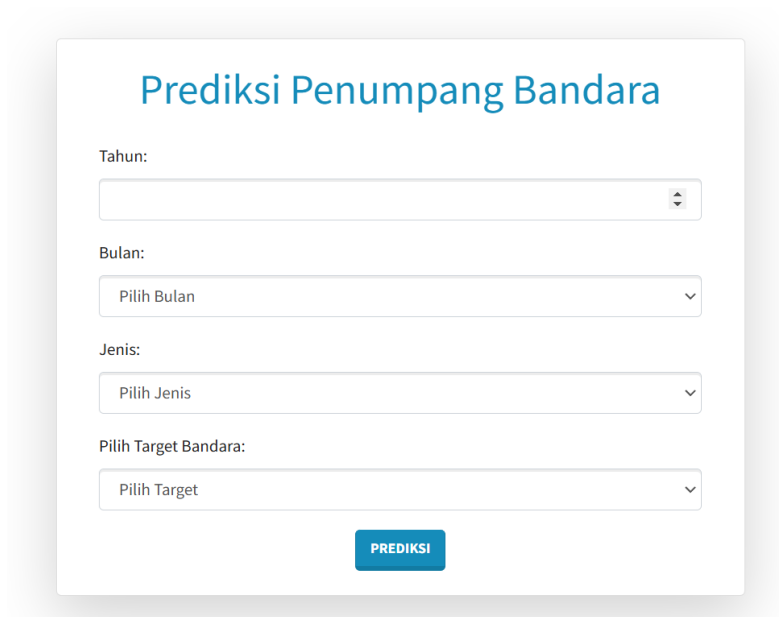
        prediction = model.predict(features_scaled)[0]
        result = int(round(prediction[targets.index(target)]))

        result = f"{result:}" .replace(".", ".")
        bulan_map = {
            1: "Januari", 2: "Februari", 3: "Maret", 4: "April", 5: "Mei", 6: "Juni",
            7: "Juli", 8: "Agustus", 9: "September", 10: "Oktober", 11: "November", 12: "Desember"
        }
        nama_bulan = bulan_map[bulan]
        jenis_map = {
            1: "Domestik",
            0: "Internasional"
        }
        jenis = jenis_map[jenis]
        return render_template('index.html', prediction=result, target=target, tahun=tahun, bulan=nama_bulan, jenis=jenis)

    return render_template('index.html', prediction=None)

```

Gambar 26 Pengimplementasian



The screenshot shows a web form titled "Prediksi Penumpang Bandara". It contains four input fields: "Tahun:" (Year), "Bulan:" (Month), "Jenis:" (Type), and "Pilih Target Bandara:" (Select Target Airport). Each field is a dropdown menu with a placeholder text "Pilih [field name]". Below the fields is a blue button labeled "PREDIKSI".

Prediksi Penumpang Bandara

Tahun:

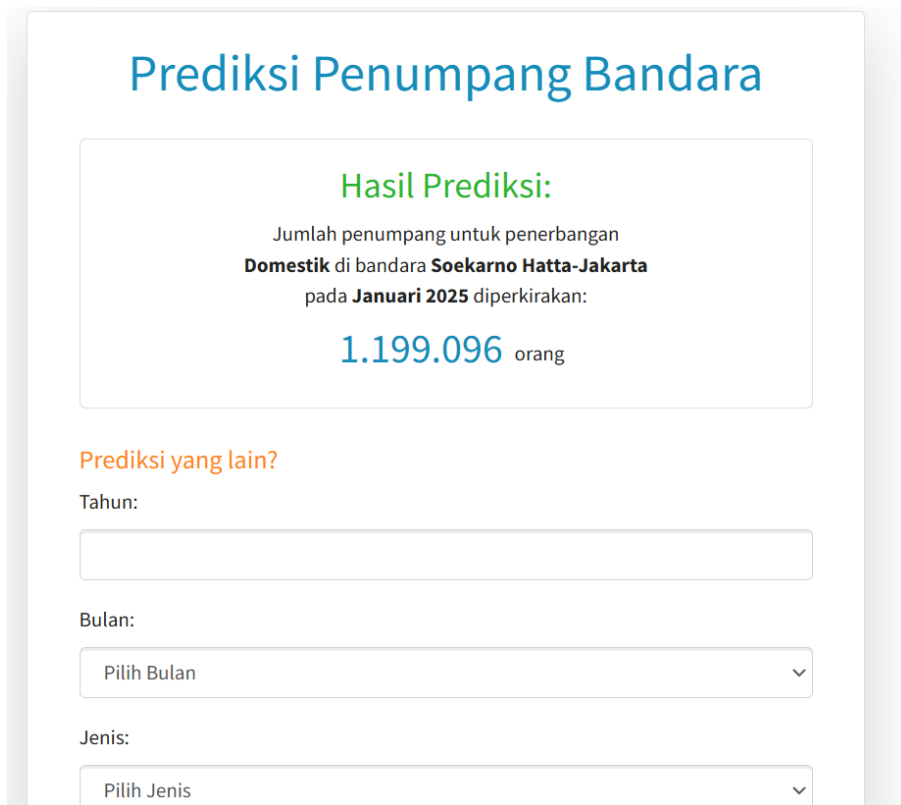
Bulan:

Jenis:

Pilih Target Bandara:

PREDIKSI

Gambar 27 Tampilan web



The screenshot shows the same web form as in Gambar 27, but with the prediction result displayed. The title "Prediksi Penumpang Bandara" is at the top. Below it, the text "Hasil Prediksi:" is in green. The prediction result is: "Jumlah penumpang untuk penerbangan Domestik di bandara Soekarno Hatta-Jakarta pada Januari 2025 diperkirakan: 1.199.096 orang". Below this, the text "Prediksi yang lain?" is in orange. The input fields for "Tahun:", "Bulan:", and "Jenis:" are still present and empty.

Prediksi Penumpang Bandara

Hasil Prediksi:

Jumlah penumpang untuk penerbangan Domestik di bandara Soekarno Hatta-Jakarta pada Januari 2025 diperkirakan: 1.199.096 orang

Prediksi yang lain?

Tahun:

Bulan:

Jenis:

Gambar 28 Hasil prediksi