

LAPORAN UJIAN TENGAH SEMESTER



PEMBELAJARAN MESIN

(PRAKTIKUM)

TI – C4

Disusun Oleh :

[434221051] | [Fairuz Zahira]

[434221053] | [Lintang Sekar Wangi]

D4 TEKNIK INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS AIRLANGGA

I. LATAR BELAKANG

Kanker otak adalah salah satu jenis kanker yang menyerang jaringan otak dan dapat berdampak serius pada sistem saraf pusat. Penyakit ini ditandai dengan pertumbuhan sel abnormal di dalam otak yang dapat mengganggu fungsi-fungsi penting seperti pengaturan gerakan, keseimbangan, dan kognisi. Meskipun kanker otak tergolong langka, deteksi dini sangat penting untuk meningkatkan peluang kesembuhan dan memperpanjang harapan hidup pasien. Namun, karena kompleksitasnya, diagnosis seringkali memerlukan analisis pemindaian medis seperti MRI atau CT scan, yang membutuhkan keahlian tinggi untuk interpretasinya.

Dalam beberapa tahun terakhir, teknologi kecerdasan buatan (AI) dan pembelajaran mesin (machine learning) semakin berkembang dan menawarkan solusi untuk deteksi dini kanker otak secara lebih cepat dan akurat. Algoritma seperti Random Forest, Decision Tree, dan Naïve Bayes dapat digunakan untuk klasifikasi dan prediksi berdasarkan data medis yang diperoleh dari hasil pemindaian. Teknologi ini memungkinkan analisis lebih efisien sehingga membantu tenaga medis dalam pengambilan keputusan klinis.

Dalam tugas ini, kelompok kami mengembangkan sebuah program deteksi kanker otak berbasis Python yang terintegrasi dengan framework Flask untuk aplikasi web. Data yang digunakan diperoleh dari platform Kaggle, dan setelah dilakukan preprocessing serta transformasi, model machine learning diterapkan untuk memprediksi keberadaan kanker otak. Tiga metode klasifikasi, yaitu Random Forest, Decision Tree, dan Naïve Bayes, dibandingkan dari segi akurasi, presisi, recall, dan F1 Score. Hasil evaluasi menunjukkan bahwa Random Forest memberikan akurasi tertinggi, sehingga dipilih sebagai metode utama dalam program ini.

Dengan adanya sistem ini, diharapkan dapat membantu proses diagnosis kanker otak secara lebih cepat dan tepat, sehingga mampu memberikan panduan yang lebih baik bagi tenaga medis dalam menentukan langkah selanjutnya untuk pengobatan. Implementasi ini berpotensi memberikan dampak signifikan dalam deteksi dini kanker otak, yang sangat krusial dalam meningkatkan peluang kesembuhan pasien.

II. HASIL

1. Link Github

<https://github.com/faerosjahera/MechineLearningUTS>

2. Penjelasan Program Secara Keseluruhan

Program deteksi kanker otak yang kami buat terdiri dari beberapa tahapan penting untuk memastikan proses deteksi berjalan dengan baik dan akurat. Data yang digunakan diambil dari platform Kaggle dan telah melalui berbagai proses pengolahan yang bertujuan untuk mempersiapkan data agar siap digunakan dalam model machine learning.

1. **Preprocessing Data:** Pada tahap ini, data dibersihkan, dinormalisasi, dan dibagi menjadi data latih dan data uji. Proses ini penting untuk memastikan data yang digunakan berkualitas, tidak ada nilai yang hilang, dan formatnya sesuai untuk diolah oleh algoritma machine learning. Proses ini juga mencakup penghilangan outlier dan penyesuaian data sesuai dengan fitur yang relevan, seperti ukuran tumor dan lokasi.
2. **Transformasi Fitur:** Data yang telah melalui preprocessing kemudian ditransformasi menjadi bentuk yang lebih mudah diproses oleh model machine learning. Teknik seperti **one-hot encoding** dan scaling data numerik digunakan untuk memastikan bahwa data memiliki format yang sesuai untuk analisis lebih lanjut.
3. **Implementasi Model Machine Learning:** Program ini menggunakan tiga metode klasifikasi, yaitu:
 - **Random Forest:** Algoritma ensemble ini menggabungkan hasil dari beberapa pohon keputusan untuk menghasilkan prediksi yang lebih akurat dan tahan terhadap overfitting.
 - **Decision Tree:** Model ini menggunakan satu pohon keputusan untuk membuat prediksi berdasarkan aturan-aturan yang didapat dari data.
 - **Naïve Bayes:** Algoritma probabilistik ini bekerja berdasarkan teorema Bayes dengan asumsi bahwa setiap fitur bersifat independen satu sama lain.
4. **Evaluasi Model:** Kinerja setiap model diukur menggunakan beberapa metrik seperti **akurasi**, **presisi**, **recall**, dan **F1 score**. Hasil evaluasi ini digunakan untuk membandingkan performa setiap model dan menentukan metode mana yang paling efektif dalam mendeteksi kanker otak. Berdasarkan hasil evaluasi, Random Forest menunjukkan akurasi tertinggi.
5. **Integrasi Web menggunakan Flask:** Program ini diintegrasikan dengan framework **Flask** untuk menyediakan antarmuka web yang memungkinkan pengguna memasukkan data, seperti hasil pemindaian MRI, dan mendapatkan prediksi kanker otak. Flask menghubungkan backend yang memproses data dengan frontend yang menampilkan hasil prediksi.

6. **Output dan Hasil Prediksi:** Hasil akhir dari program ini berupa prediksi apakah seseorang berisiko terkena kanker otak. Prediksi tersebut ditampilkan melalui antarmuka web yang telah dirancang, sehingga pengguna dapat dengan mudah mengakses informasi yang diperlukan. Selain prediksi, informasi tambahan terkait probabilitas diagnosis juga ditampilkan untuk membantu dokter dalam pengambilan keputusan medis.

Dengan memanfaatkan pustaka Python seperti **scikit-learn** untuk machine learning dan **pandas** serta **numpy** untuk manipulasi data, program ini mampu mengolah dataset kanker otak secara efisien dan memberikan prediksi yang dapat diandalkan. Kombinasi dari model machine learning dan antarmuka web berbasis Flask ini bertujuan untuk memberikan solusi yang cepat dan akurat dalam mendeteksi kanker otak.

3. Hasil

Metode Decision Tree

Membaca Data dan Mengecek Missing Value

```
brain_tumor_path = 'Brain Tumor.csv'  
brain_tumor_df = pd.read_csv(brain_tumor_path)  
df = pd.read_csv('Brain Tumor.csv')  
print(df.isna().sum())
```

Kode brain_tumor_df diatas berfungsi untuk membaca data dari csv.
fungsi isnan() untuk mengecek apakah ada missing value di data yang kita pakai.
fungsi sum() untuk menghitung berapa banyak jumlah data yang missing value.

Outputnya

Pada semua kolom tidak terdeteksi adanya missing value

Image	0
Class	0
Mean	0
Variance	0
Standard Deviation	0
Entropy	0
Skewness	0
Kurtosis	0
Contrast	0
Energy	0
ASM	0
Homogeneity	0
Dissimilarity	0
Correlation	0
Coarseness	0

Memilih kolom numerik

```

numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
numeric_cols.remove('Class')

print("Kolom numerik yang digunakan:", numeric_cols)

```

Kode di atas bertujuan untuk digunakan untuk memisahkan jenis kolom yang berisi nilai numerik dan huruf, karena di dataset saya terdapat kolom yang berisikan huruf. Pada kode diatas juga saya memisahkan kolom “Class” karena berisi hasil hasil prediksi kanker otak.

Output

```

Kolom numerik yang digunakan: ['Mean', 'Variance', 'Standard Deviation', 'Entropy', 'Skewness', 'Kurtosis', 'Contrast', 'Energy', 'ASM', 'Homogeneity', 'Dissimilarity', 'Correlation', 'Coarseness']

```

Mencari Outlier dan Penanganannya

```

z_scores = np.abs(stats.zscore(df[numeric_cols]))
threshold = 3
outliers = (z_scores > threshold).any(axis=1)

# Menampilkan jumlah outlier
print(f"Jumlah outlier yang terdeteksi: {outliers.sum()}")

# Menampilkan baris yang dianggap outlier
df_outliers = df[outliers]
print("Data outlier:")
print(df_outliers)

# Menghapus baris outlier dari dataset
df_clean = df[~outliers]
print(f"Dataset setelah menghapus outlier: {df_clean.shape}")
| 

# Menghitung ulang Z-Score setelah penanganan outlier
z_scores_clean = np.abs(stats.zscore(df_clean[numeric_cols]))
outliers_clean = (z_scores_clean > threshold).any(axis=1)
print(f"Jumlah outlier setelah penanganan: {outliers_clean.sum()}")

```

Di Bagian ini saya melakukan pencari outlier di seluruh dataset yang saya gunakan dengan variabel numeric cols / variabel yang sudah saya pisahkan sebelumnya. Setelah menemukan outlier saya menangani nya dengan cara menghapus semua outlier yang ada dengan syntax ~ yang berarti mengecualikan. Lalu setelah outlier dihapus saya menghitung ulang lagi apakah masih ada outlier yang tersisa setelah saya hapus sebelumnya.

Output

Pada awal deteksi outlier, kami menemukan 3762 data outlier lalu setelah kita tangani dan melakukan pengecekan ulang data outlier menjadi 0

	Image	Class	Mean	Variance	Standard Deviation	...	ASM	Homogeneity	Dissimilarity	Correlation	Coarseness
0	Image1	0	6.535339	619.587845	24.891522	...	0.086033	0.530941	4.473346	0.981939	7.458341e-155
1	Image2	0	8.749969	805.957634	28.389393	...	0.225674	0.651352	3.220072	0.988834	7.458341e-155
2	Image3	1	7.341095	1143.808219	33.820234	...	0.001019	0.268275	5.981800	0.978014	7.458341e-155
3	Image4	1	5.958145	959.711985	30.979219	...	0.001026	0.243851	7.700919	0.964189	7.458341e-155
4	Image5	0	7.315231	729.540579	27.010009	...	0.118232	0.501140	6.834689	0.972789	7.458341e-155
...
3757	Image3758	0	21.234512	1208.850174	34.768523	...	0.048693	0.487131	5.211739	0.950972	7.458341e-155
3758	Image3759	0	20.435349	1227.151440	35.030721	...	0.051045	0.502712	5.083126	0.952749	7.458341e-155
3759	Image3760	0	18.011520	1151.582765	33.934978	...	0.052409	0.492269	5.103700	0.952181	7.458341e-155
3760	Image3761	0	13.330429	945.732779	30.752769	...	0.068397	0.480064	6.439784	0.940698	7.458341e-155
3761	Image3762	0	6.110138	480.884025	21.929068	...	0.093773	0.494333	6.787329	0.938731	7.458341e-155

```

Dataset setelah menghapus outlier: (0, 15)
Jumlah outlier setelah penanganan: 0

```

Normalisasi Z-Score

```

# Z-Score
print("Normalisasi Z Score:")

df['Mean'] = (df['Mean'] - df['Mean'].mean()) / df['Mean'].std()
df['Variance_Z'] = (df['Variance'] - df['Variance'].mean()) / df['Variance'].std()
df['Entropy'] = (df['Entropy'] - df['Entropy'].mean()) / df['Entropy'].std()
df['Skewness'] = (df['Skewness'] - df['Skewness'].mean()) / df['Skewness'].std()
df['Kurtosis'] = (df['Kurtosis'] - df['Kurtosis'].mean()) / df['Kurtosis'].std()
df['Contrast'] = (df['Contrast'] - df['Contrast'].mean()) / df['Contrast'].std()
df['Energy'] = (df['Energy'] - df['Energy'].mean()) / df['Energy'].std()
df['ASM'] = (df['ASM'] - df['ASM'].mean()) / df['ASM'].std()
df['Homogeneity'] = (df['Homogeneity'] - df['Homogeneity'].mean()) / df['Homogeneity'].std()
df['Dissimilarity'] = (df['Dissimilarity'] - df['Dissimilarity'].mean()) / df['Dissimilarity'].std()
df['Correlation'] = (df['Correlation'] - df['Correlation'].mean()) / df['Correlation'].std()
df['Coarseness'] = (df['Coarseness'] - df['Coarseness'].mean()) / df['Coarseness'].std()

print(df[['Mean', 'Variance', 'Entropy', 'Skewness', 'Kurtosis', 'Contrast', 'Energy', 'ASM', 'Homogeneity', 'Dissimilarity', 'Correlation', 'Coarseness']])

```

Normalisasi Z-Score dilakukan dengan cara mengurangi kolom data asli dengan rata-rata kolom itu sendiri lalu dibagi dengan standar deviasi pada kolom itu sendiri. Normalisasi ini dilakukan pada tiap kolom fitur/ kolom numerik yang sudah saya pisah diatas sebelumnya.

Output

Normalisasi Z Score:												
	Mean	Variance	Entropy	Skewness	Kurtosis	...	ASM	Homogeneity	Dissimilarity	Correlation	Coarseness	...
0	-0.515632	619.587845	0.504583	0.067846	-0.097254	...	0.470011	0.404046	-0.121692	1.000580	-inf	...
1	-0.129001	805.957634	2.745685	-0.150184	-0.175857	...	2.865199	1.345278	-0.799075	1.264209	-inf	...
2	-0.374963	1143.888219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	...
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	...
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	...
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	...
1	-0.129001	805.957634	2.745685	-0.150184	-0.175857	...	2.865199	1.345278	-0.799075	1.264209	-inf	...
2	-0.374963	1143.888219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	...
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	...
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	...
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	...
2	-0.374963	1143.888219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	...
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	...
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	...
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	...
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	...
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	...
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	...
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	...
3758	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.207888	-0.115361	inf	...
3758	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.207888	-0.115361	inf	...
3758	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.207888	-0.115361	inf	...

Grouping Data

```

# Preparing the dataset
X = brain_tumor_df.drop(['Image', 'Class'], axis=1) # Features
y = brain_tumor_df['Class'] # Target (Class)

```

Disini variabel X akan berisi faktor-faktor atau fitur yang mempengaruhi data, disini saya mengambil semua data pada tabel kecuali kolom "Image" dan "Class".

axis = 1 disitu berguna untuk menghapus kolom, jika = 0 menghapus baris

Sedangkan pada variabel Y berisi kolom target yaitu kolom "Class"

Splitting Data

```

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

Pada bagian ini data akan dipisahkan menjadi data training dan data testing. Disini akan dipisahkan menjadi 80% data training dan 20% data testing.

Decision Tree

```
# Creating and training the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

clf.fit(X_train, y_train): Melatih (fit) model pohon keputusan dengan data latih (X_train) dan label (y_train). Ini adalah langkah di mana model belajar dari data.

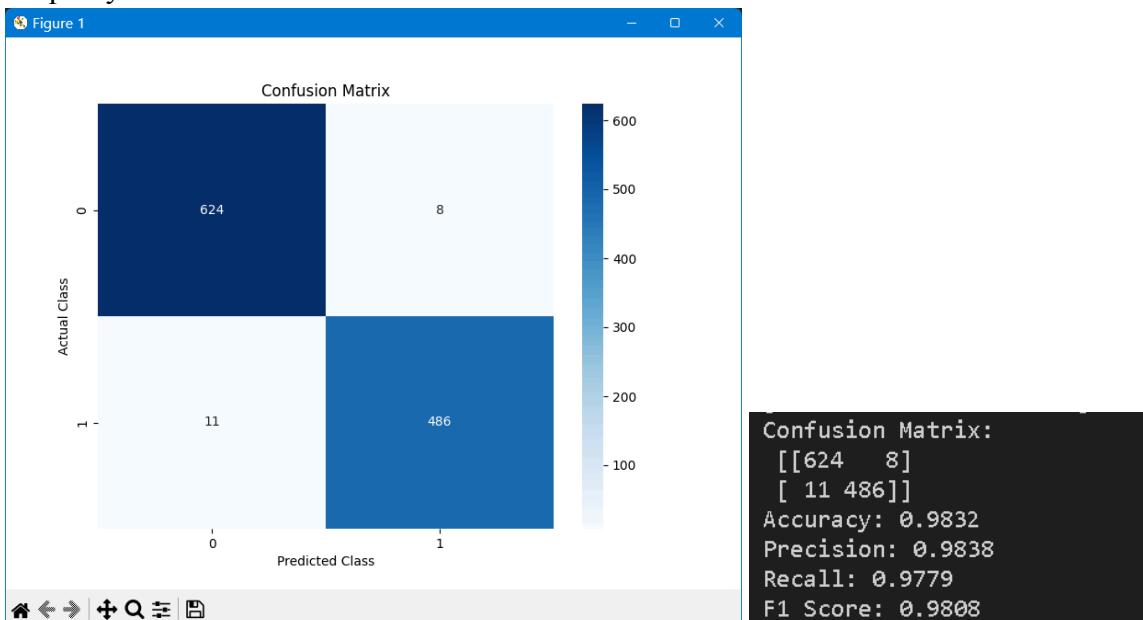
clf.predict(X_test): Menggunakan model yang telah dilatih untuk memprediksi nilai target dari data uji (X_test). Hasil prediksi disimpan dalam y_pred.

Mengecek Akurasi

```
# Evaluating the model using confusion matrix and calculating performance metrics
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Disini kami menggunakan library yang sudah disediakan python untuk pengecekan confusion matrix, akurasi, presisi, recall, dan f1

Outputnya



Hasil dari Figure Decision Tree

```
# Visualizing the decision tree
plt.figure(figsize=(12, 8))
tree.plot_tree(dt_classifier, feature_names=X.columns, class_names=["Non-Tumor", "Tumor"], filled=True)
plt.title("Decision Tree Visualization")
plt.show()
```

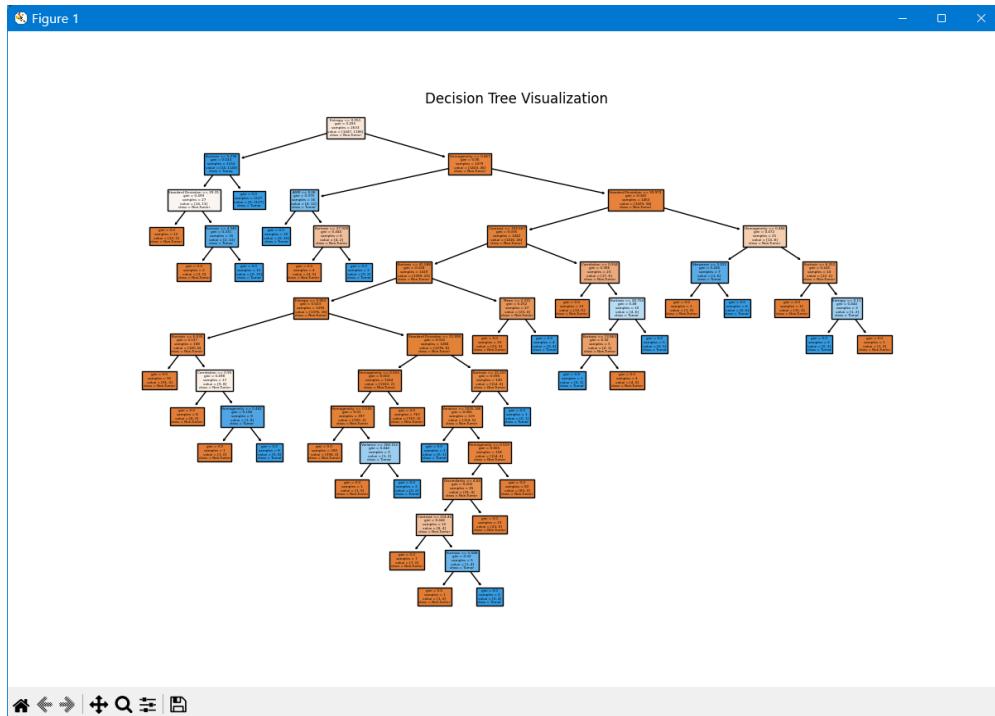
tree.plot_tree(...): Menggunakan fungsi plot_tree untuk menggambar pohon keputusan. Parameter yang digunakan:

clf: Model yang sudah dilatih.

feature_names=X.columns: Nama-nama fitur yang digunakan dalam pohon keputusan.

class_names=np.unique(y).astype(str): Nama-nama kelas yang akan ditampilkan.

filled=True: Mengisi warna pada node berdasarkan kelas, sehingga visualisasi menjadi lebih informatif.



Metode Naive Bayes

Membaca Data dan Mengecek Missing Value

```
brain_tumor_path = 'Brain Tumor.csv'  
brain_tumor_df = pd.read_csv(brain_tumor_path)  
df = pd.read_csv('Brain Tumor.csv')  
print(df.isna().sum())
```

Kode brain_tumor_df diatas berfungsi untuk membaca data dari csv.

fungsi isnull() untuk mengecek apakah ada missing value di data yang kita pakai.

fungsi sum() untuk menghitung berapa banyak jumlah data yang missing value.

Outputnya

Pada semua kolom tidak terdeteksi adanya missing value

```
Image          0
Class          0
Mean           0
Variance       0
Standard Deviation 0
Entropy         0
Skewness        0
Kurtosis        0
Contrast        0
Energy          0
ASM             0
Homogeneity     0
Dissimilarity   0
Correlation     0
Coarseness      0
```

Memilih kolom numerik

```
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
numeric_cols.remove('Class')

print("Kolom numerik yang digunakan:", numeric_cols)
```

Kode di atas bertujuan untuk digunakan untuk memisahkan jenis kolom yang berisi nilai numerik dan huruf, karena di dataset saya terdapat kolom yang berisikan huruf. Pada kode diatas juga saya memisahkan kolom “Class” karena berisi hasil prediksi kanker otak.

Output

```
Kolom numerik yang digunakan: ['Mean', 'Variance', 'Standard Deviation', 'Entropy', 'Skewness', 'Kurtosis', 'Contrast', 'Energy', 'ASM', 'Homogeneity', 'Dissimilarity', 'Correlation', 'Coarseness']
```

Mencari Outlier dan Penanganannya

```

z_scores = np.abs(stats.zscore(df[numerical_cols]))
threshold = 3
outliers = (z_scores > threshold).any(axis=1)

# Menampilkan jumlah outlier
print(f"Jumlah outlier yang terdeteksi: {outliers.sum()}\n")

# Menampilkan baris yang dianggap outlier
df_outliers = df[outliers]
print("Data outlier:")
print(df_outliers)

# Menghapus baris outlier dari dataset
df_clean = df[~outliers]
print(f"Dataset setelah menghapus outlier: {df_clean.shape}\n")

# Menghitung ulang Z-Score setelah penanganan outlier
z_scores_clean = np.abs(stats.zscore(df_clean[numerical_cols]))
outliers_clean = (z_scores_clean > threshold).any(axis=1)
print(f"Jumlah outlier setelah penanganan: {outliers_clean.sum()}")

```

Di Bagian ini saya melakukan pencari outlier di seluruh dataset yang saya gunakan dengan variabel numeric cols / variabel yang sudah saya pisahkan sebelumnya. Setelah menemukan outlier saya menangani nya dengan cara menghapus semua outlier yang ada dengan syntax ~ yang berarti mengecualikan. Lalu setelah outlier dihapus saya menghitung ulang lagi apakah masih ada outlier yang tersisa setelah saya hapus sebelumnya.

Output

Pada awal deteksi outlier, kami menemukan 3762 data outlier lalu setelah kita tangani dan melakukan pengecekan ulang data outlier menjadi 0

Jumlah outlier yang terdeteksi: 3762												
Data outlier:												
	Image	Class	Mean	Variance	Standard Deviation	...	ASM	Homogeneity	Dissimilarity	Correlation	Coarseness	
0	Image1	0	6.535339	619.587845	24.891522	...	0.086033	0.530941	4.473346	0.981939	7.458341e-155	
1	Image2	0	8.749969	805.957634	28.389393	...	0.225674	0.651352	3.220072	0.988834	7.458341e-155	
2	Image3	1	7.341095	1143.808219	33.820234	...	0.001019	0.268275	5.981800	0.978014	7.458341e-155	
3	Image4	1	5.958145	959.711985	30.979219	...	0.001026	0.243851	7.700919	0.964189	7.458341e-155	
4	Image5	0	7.315231	729.540579	27.010009	...	0.118232	0.501140	6.834689	0.972789	7.458341e-155	
...	
3757	Image3758	0	21.234512	1208.850174	34.768523	...	0.048693	0.487131	5.211739	0.950972	7.458341e-155	
3758	Image3759	0	20.435349	1227.151440	35.030721	...	0.051045	0.502712	5.083126	0.952749	7.458341e-155	
3759	Image3760	0	18.011520	1151.582765	33.934978	...	0.052409	0.492269	5.103700	0.952181	7.458341e-155	
3760	Image3761	0	13.330429	945.732779	30.752769	...	0.068397	0.480064	6.439784	0.940898	7.458341e-155	
3761	Image3762	0	6.110138	480.884025	21.929068	...	0.093773	0.494333	6.787329	0.938731	7.458341e-155	

Dataset setelah menghapus outlier: (0, 15)
Jumlah outlier setelah penanganan: 0

Normalisasi Z-Score

```

# Z-Score
print("\nNormalisasi Z Score:")

df['Mean'] = (df['Mean'] - df['Mean'].mean()) / df['Mean'].std()
df['Variance_Z'] = (df['Variance'] - df['Variance'].mean()) / df['Variance'].std()
df['Entropy'] = (df['Entropy'] - df['Entropy'].mean()) / df['Entropy'].std()
df['Skewness'] = (df['Skewness'] - df['Skewness'].mean()) / df['Skewness'].std()
df['Kurtosis'] = (df['Kurtosis'] - df['Kurtosis'].mean()) / df['Kurtosis'].std()
df['Contrast'] = (df['Contrast'] - df['Contrast'].mean()) / df['Contrast'].std()
df['Energy'] = (df['Energy'] - df['Energy'].mean()) / df['Energy'].std()
df['ASM'] = (df['ASM'] - df['ASM'].mean()) / df['ASM'].std()
df['Homogeneity'] = (df['Homogeneity'] - df['Homogeneity'].mean()) / df['Homogeneity'].std()
df['Dissimilarity'] = (df['Dissimilarity'] - df['Dissimilarity'].mean()) / df['Dissimilarity'].std()
df['Correlation'] = (df['Correlation'] - df['Correlation'].mean()) / df['Correlation'].std()
df['Coarseness'] = (df['Coarseness'] - df['Coarseness'].mean()) / df['Coarseness'].std()

print(df[['Mean', 'Variance', 'Entropy', 'Skewness', 'Kurtosis', 'Contrast', 'Energy', 'ASM', 'Homogeneity', 'Dissimilarity', 'Correlation', 'Coarseness']])

```

Normalisasi Z-Score dilakukan dengan cara mengurangi kolom data asli dengan rata-rata kolom itu sendiri lalu dibagi dengan standar deviasi pada kolom itu sendiri. Normalisasi ini dilakukan pada tiap kolom fitur/ kolom numerik yang sudah saya pisah diatas sebelumnya.

Output

Normalisasi Z Score:												
	Mean	Variance	Entropy	Skewness	Kurtosis	...	ASM	Homogeneity	Dissimilarity	Correlation	Coarseness	
0	-0.515632	619.587845	0.504583	0.067846	-0.097254	...	0.470011	0.404046	-0.121692	1.000580	-inf	
1	-0.129001	805.957634	2.745685	-0.150184	-0.175857	...	2.865199	1.345278	-0.799075	1.264209	-inf	
2	-0.374963	1143.808219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
1	-0.129001	805.957634	2.745685	-0.150184	-0.175857	...	2.865199	1.345278	-0.799075	1.264209	-inf	
2	-0.374963	1143.808219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
2	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
3758	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.267888	-0.115361	inf	
3758	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.267888	-0.115361	inf	
3759	-1.497884	1151.592765	0.371090	0.700573	0.323209	...	0.105737	0.101750	0.240000	0.127077	inf	

Grouping Data

```
# Preparing the dataset
X = brain_tumor_df.drop(['Image', 'Class'], axis=1) # Features
y = brain_tumor_df['Class'] # Target (Class)
```

Disini variabel X akan berisi faktor-faktor atau fitur yang mempengaruhi data, disini saya mengambil semua data pada tabel kecuali kolom “Image” dan “Class”.
axis = 1 disitu berguna untuk menghapus kolom, jika = 0 menghapus baris
Sedangkan pada variabel Y berisi kolom target yaitu kolom “Class”

Splitting Data

```
# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Pada bagian ini data akan dipisahkan menjadi data training dan data testing. Disini akan dipisahkan menjadi 80% data training dan 20% data testing.

Naive Bayes

```

# Initialize and train the Naive Bayes classifier
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)

# Make predictions
y_pred = nb_classifier.predict(X_test)
print("instance prediksi naive bayes:")
print (y_pred)

```

GaussianNB() adalah fungsi dari scikit-learn untuk membuat model Naive Bayes Gaussian. fit() digunakan untuk melatih (train) model Naive Bayes menggunakan data training.

X_train adalah fitur-fitur yang digunakan untuk melatih model, dan y_train adalah label (target) dari data training.

predict() digunakan untuk memprediksi hasil dari data testing berdasarkan model yang sudah dilatih.

X_test adalah fitur-fitur yang digunakan untuk menguji model, dan hasil prediksinya disimpan dalam variabel y_pred.

accuracy_score() menghitung akurasi model dengan membandingkan hasil prediksi (y_pred) dengan nilai sebenarnya (y_test).

Output

```

instance prediksi naive bayes:
[1 0 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0
 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 1 0 0 1 0 1 1 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0
 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 1 1 1 1 1
 1 0 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 1 0 0 1 1 0 0
 0 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 1 1 0 0 1 0
 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 0 0 1 1 0 1
 0 1 0 0 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 1 0 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0
 0 1 1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0
 0 1 1 1 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0
 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1
 0 1 0 1 1 0 1 0 0 1 1 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0
 0 0 1 1 1 0 0 1 0 1 1 1 0 0 0 1 0 1 1 1 0 0 1 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0
 0 0 1 1 1 1 0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0 1 0
 1 0 1 1 0 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 0
 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0
 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0 0 0 1
 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0
 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1
 1 0 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0
 1 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0]

```

Mengecek Akurasi

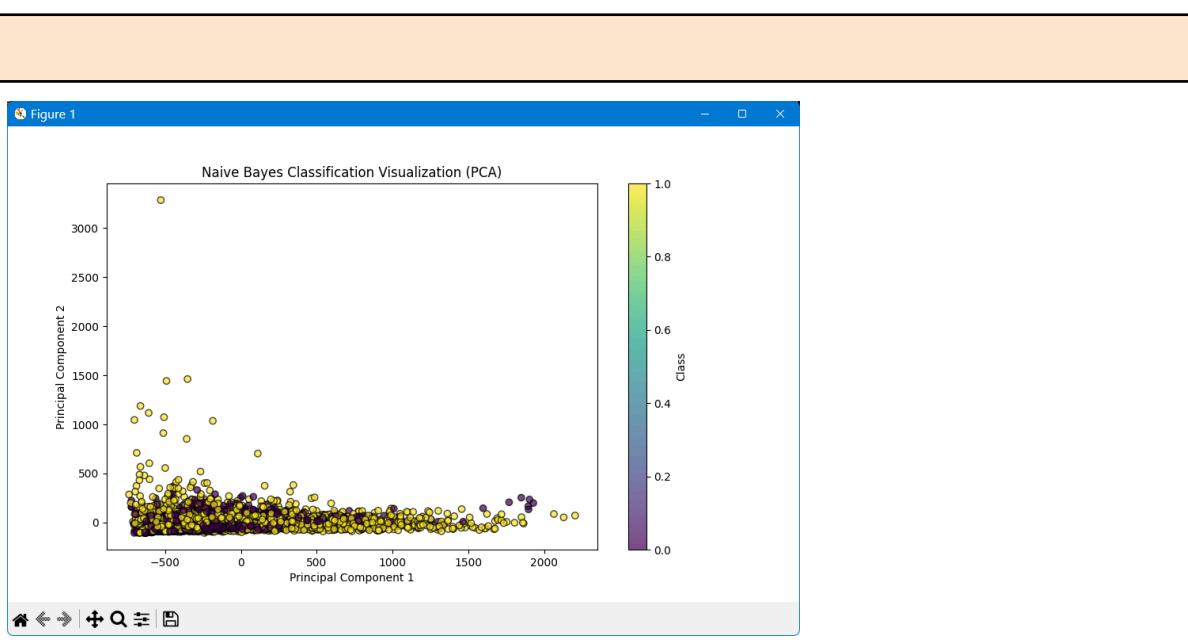
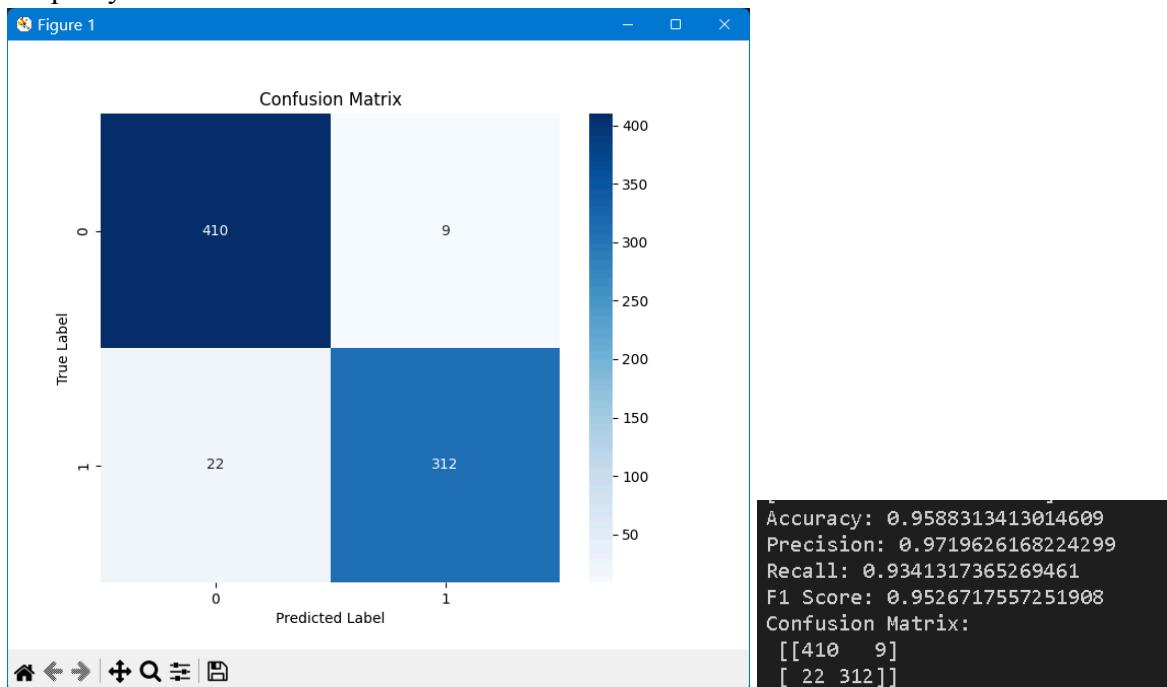
```

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

```

Disini kami menggunakan library yang sudah disediakan python untuk pengecekan confusion matrix, akurasi, presisi, recall, dan f1

Outputnya



Metode Random Forest

Membaca Data dan Mengecek Missing Value

```
brain_tumor_path = 'Brain Tumor.csv'  
brain_tumor_df = pd.read_csv(brain_tumor_path)  
df = pd.read_csv('Brain Tumor.csv')  
print(df.isna().sum())
```

Kode brain_tumor_df diatas berfungsi untuk membaca data dari csv.
fungsi isnull() untuk mengecek apakah ada missing value di data yang kita pakai.
fungsi sum() untuk menghitung berapa banyak jumlah data yang missing value.

Outputnya

Pada semua kolom tidak terdeteksi adanya missing value

```
Image          0  
Class          0  
Mean           0  
Variance       0  
Standard Deviation  0  
Entropy         0  
Skewness        0  
Kurtosis        0  
Contrast        0  
Energy          0  
ASM             0  
Homogeneity     0  
Dissimilarity   0  
Correlation      0  
Coarseness       0
```

Memilih kolom numerik

```
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()  
numeric_cols.remove('Class')  
  
print("Kolom numerik yang digunakan:", numeric_cols)
```

Kode di atas bertujuan untuk digunakan untuk memisahkan jenis kolom yang berisi nilai numerik dan huruf, karena di dataset saya terdapat kolom yang berisikan huruf. Pada kode diatas juga saya memisahkan kolom “Class” karena berisi hasil prediksi kanker otak.

Output

```
Kolom numerik yang digunakan: ['Mean', 'Variance', 'Standard Deviation', 'Entropy', 'Skewness', 'Kurtosis', 'Contrast', 'Energy', 'ASM', 'Homogeneity', 'Dissimilarity', 'Correlation', 'Coarseness']
```

Mencari Outlier dan Penanganannya

```
z_scores = np.abs(stats.zscore(df[numeric_cols]))
threshold = 3
outliers = (z_scores > threshold).any(axis=1)

# Menampilkan jumlah outlier
print(f"Jumlah outlier yang terdeteksi: {outliers.sum()}")

# Menampilkan baris yang dianggap outlier
df_outliers = df[outliers]
print("Data outlier:")
print(df_outliers)

# Menghapus baris outlier dari dataset
df_clean = df[~outliers]
print(f"Dataset setelah menghapus outlier: {df_clean.shape}")
| 
# Menghitung ulang Z-Score setelah penanganan outlier
z_scores_clean = np.abs(stats.zscore(df_clean[numeric_cols]))
outliers_clean = (z_scores_clean > threshold).any(axis=1)
print(f"Jumlah outlier setelah penanganan: {outliers_clean.sum()}")
```

Di Bagian ini saya melakukan pencari outlier di seluruh dataset yang saya gunakan dengan variabel numeric cols / variabel yang sudah saya pisahkan sebelumnya. Setelah menemukan outlier saya menangani nya dengan cara menghapus semua outlier yang ada dengan syntax ~ yang berarti mengecualikan. Lalu setelah outlier dihapus saya menghitung ulang lagi apakah masih ada outlier yang tersisa setelah saya hapus sebelumnya.

Output

Pada awal deteksi outlier, kami menemukan 3762 data outlier lalu setelah kita tangani dan melakukan pengecekan ulang data outlier menjadi 0

```
Jumlah outlier yang terdeteksi: 3762
Data outlier:
   Image  Class      Mean    Variance  Standard Deviation ...      ASM  Homogeneity  Dissimilarity  Correlation  Coarseness
0  Image1     0  6.535339  619.587845       24.891522 ...  0.086033  0.530941  4.473346  0.981939  7.458341e-155
1  Image2     0  8.749969  805.957634       28.389393 ...  0.225674  0.651352  3.220072  0.988834  7.458341e-155
2  Image3     1  7.341095  1143.808219       33.820234 ...  0.001019  0.268275  5.981800  0.978014  7.458341e-155
3  Image4     1  5.958145  959.711985       30.979219 ...  0.001026  0.243851  7.700919  0.964189  7.458341e-155
4  Image5     0  7.315231  729.540579       27.010009 ...  0.118232  0.501140  6.834689  0.972789  7.458341e-155
...  ...
3757 Image3758     0  21.234512  1208.850174       34.768523 ...  0.048693  0.487131  5.211739  0.950972  7.458341e-155
3758 Image3759     0  20.435349  1227.151440       35.030721 ...  0.051045  0.502712  5.083126  0.952749  7.458341e-155
3759 Image3760     0  18.011520  1151.582765       33.934978 ...  0.052409  0.492269  5.103700  0.952181  7.458341e-155
3760 Image3761     0  13.330429  945.732779       30.752769 ...  0.068397  0.480064  6.439784  0.940898  7.458341e-155
3761 Image3762     0  6.110138  480.884025       21.929068 ...  0.093773  0.494333  6.787329  0.938731  7.458341e-155
```

```
Dataset setelah menghapus outlier: (0, 15)
Jumlah outlier setelah penanganan: 0
```

Normalisasi Z-Score

```
# Z-Score
print("\nNormalisasi Z Score:")

df['Mean'] = (df['Mean'] - df['Mean'].mean()) / df['Mean'].std()
df['Variance_Z'] = (df['Variance'] - df['Variance'].mean()) / df['Variance'].std()
df['Entropy'] = (df['Entropy'] - df['Entropy'].mean()) / df['Entropy'].std()
df['Skewness'] = (df['Skewness'] - df['Skewness'].mean()) / df['Skewness'].std()
df['Kurtosis'] = (df['Kurtosis'] - df['Kurtosis'].mean()) / df['Kurtosis'].std()
df['Contrast'] = (df['Contrast'] - df['Contrast'].mean()) / df['Contrast'].std()
df['Energy'] = (df['Energy'] - df['Energy'].mean()) / df['Energy'].std()
df['ASM'] = (df['ASM'] - df['ASM'].mean()) / df['ASM'].std()
df['Homogeneity'] = (df['Homogeneity'] - df['Homogeneity'].mean()) / df['Homogeneity'].std()
df['Dissimilarity'] = (df['Dissimilarity'] - df['Dissimilarity'].mean()) / df['Dissimilarity'].std()
df['Correlation'] = (df['Correlation'] - df['Correlation'].mean()) / df['Correlation'].std()
df['Coarseness'] = (df['Coarseness'] - df['Coarseness'].mean()) / df['Coarseness'].std()

print(df[['Mean', 'Variance', 'Entropy', 'Skewness', 'Kurtosis', 'Contrast', 'Energy', 'ASM', 'Homogeneity', 'Dissimilarity', 'Correlation', 'Coarseness']])
```

Normalisasi Z-Score dilakukan dengan cara mengurangi kolom data asli dengan rata-rata kolom itu sendiri lalu dibagi dengan standar deviasi pada kolom itu sendiri. Normalisasi ini dilakukan pada tiap kolom fitur/ kolom numerik yang sudah saya pisah diatas sebelumnya.

Output

Normalisasi Z Score:												
	Mean	Variance	Entropy	Skewness	Kurtosis	...	ASM	Homogeneity	Dissimilarity	Correlation	Coarseness	
0	-0.515632	619.587845	0.504583	0.067846	-0.097254	...	0.470011	0.404046	-0.121692	1.000580	-inf	
1	-0.129001	805.957634	2.745685	-0.150184	-0.175857	...	2.865199	1.345278	-0.799075	1.264209	-inf	
2	-0.374963	1143.808219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
1	-0.129001	805.957634	2.745685	-0.150184	-0.175857	...	2.865199	1.345278	-0.799075	1.264209	-inf	
2	-0.374963	1143.808219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
2	-0.374963	1143.808219	-1.026571	0.374481	0.037043	...	-0.988209	-1.649172	0.693612	0.850523	-inf	
3	-0.616399	959.711985	-1.026424	0.615106	0.160181	...	-0.988091	-1.840090	1.622779	0.321998	-inf	
4	-0.379478	729.540579	1.041118	0.070480	-0.094090	...	1.022306	0.171092	1.154590	0.650767	-inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
...
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
3757	2.050555	1208.850174	-0.139873	-0.789026	-0.349816	...	-0.170465	0.061588	0.277402	-0.183311	inf	
3758	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.207888	-0.115361	inf	
3758	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.207888	-0.115361	inf	
3759	1.911036	1227.151440	-0.097332	-0.764603	-0.345656	...	-0.130133	0.183384	0.207888	-0.115361	inf	

Grouping Data

```
# Preparing the dataset
X = brain_tumor_df.drop(['Image', 'Class'], axis=1) # Features
y = brain_tumor_df['Class'] # Target (Class)
```

Disini variabel X akan berisi faktor-faktor atau fitur yang mempengaruhi data, disini saya mengambil semua data pada tabel kecuali kolom “Image” dan “Class”.

axis = 1 disitu berguna untuk menghapus kolom, jika = 0 menghapus baris

Sedangkan pada variabel Y berisi kolom target yaitu kolom “Class”

Splitting Data

```
# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Pada bagian ini data akan dipisahkan menjadi data training dan data testing. Disini akan pisahkan menjadi 80% data training dan 20% data testing.

Random Forest

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)
```

rf_model.fit(X_train, y_train): Melatih (fit) model pohon keputusan dengan data latih (X_train) dan label (y_train). Ini adalah langkah di mana model belajar dari data.

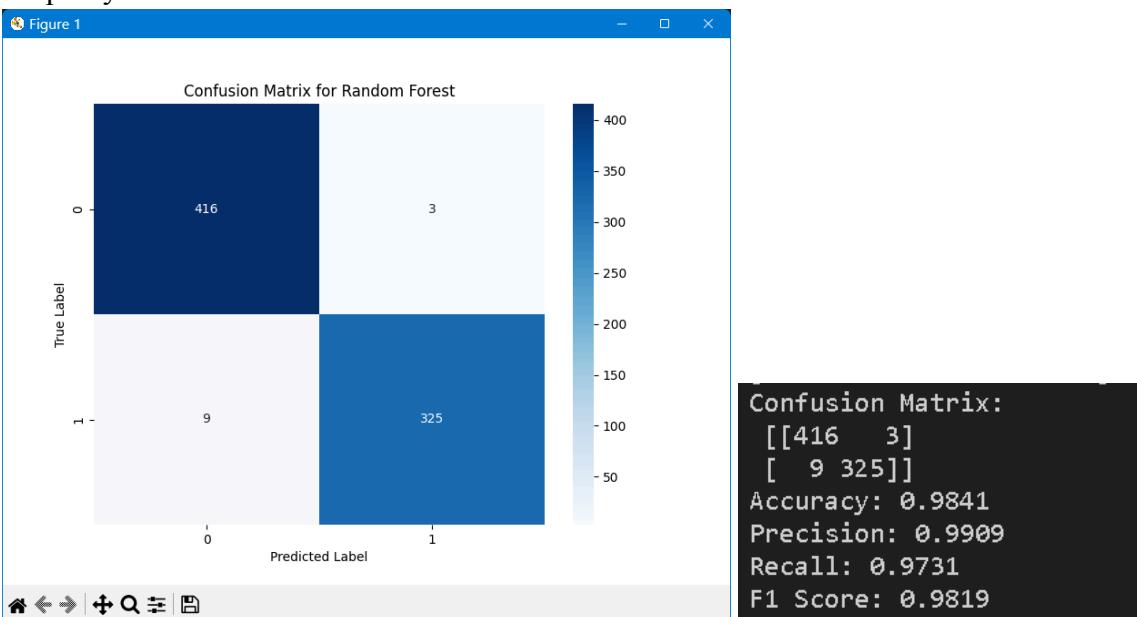
rf_model.predict(X_test): Menggunakan model yang telah dilatih untuk memprediksi nilai target dari data uji (X_test). Hasil prediksi disimpan dalam y_pred.

Mengecek Akurasi

```
# Calculate evaluation metrics
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Disini kami menggunakan library yang sudah disediakan python untuk pengecekan confusion matrix, akurasi, presisi, recall, dan f1

Outputnya



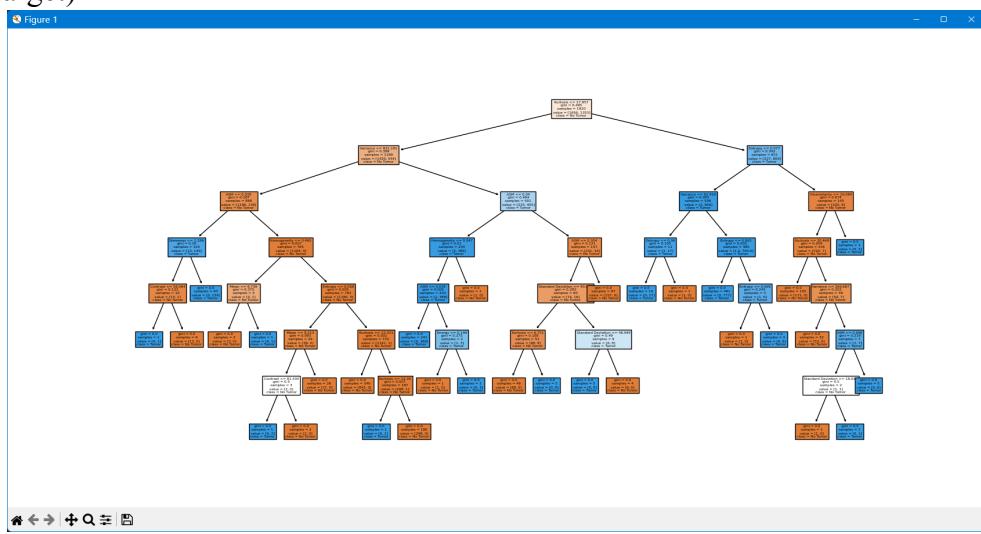
Hasil Dalam Figure Random Forest

```
# Visualize one of the decision trees in the Random Forest
estimator = rf_model.estimators_[0]
fig = plt.figure(figsize=(25, 20))
_ = tree.plot_tree(estimator,
                   feature_names=X.columns,
                   class_names=["No Tumor", "Tumor"],
                   filled=True)
plt.savefig('random_forest_tree.png')
plt.show()
```

rf_model.estimators_ berisi semua decision tree yang dibangun selama pelatihan, dan [0] berarti mengambil decision tree pertama.

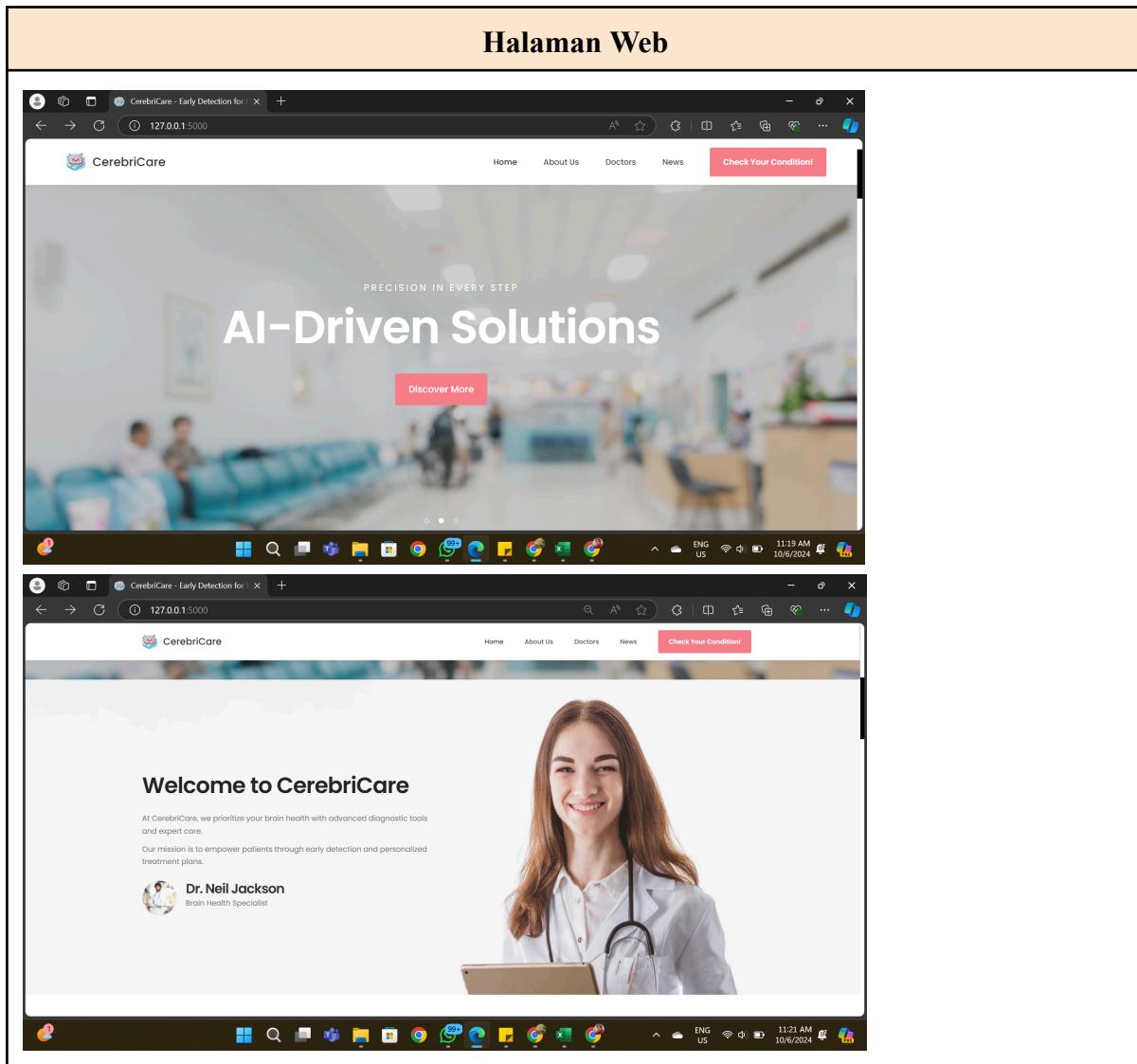
feature_names=X.columns digunakan untuk memberikan label nama fitur (yaitu kolom dari dataset), dan class_names=["No Tumor", "Tumor"] memberi label pada class prediksi

(kelas target)



Output dan Hasil Prediksi

Dari hasil perbandingan tiga metode (Random Forest, Decision Tree, dan Naïve PCA), **Random Forest** menunjukkan performa terbaik dengan **Accuracy** tertinggi (0.9841), **Precision** tertinggi (0.9909), dan **F1 Score** tertinggi (0.9819). Meskipun **Recall** pada **Decision Tree** sedikit lebih tinggi (0.9779 dibandingkan dengan 0.9731 pada Random Forest), secara keseluruhan Random Forest tetap menjadi metode yang paling unggul dalam hal keseimbangan metrik. Jadi kami mengaplikasikan dalam bentuk web (flask) menggunakan metode **Random Forest**.



CerebriCare - Early Detection for Brain Cancer

127.0.0.1:5000

CerebriCare

Home About Us Doctors News Check Your Condition

Our Doctors

Dr. Thompson Baston
Neurosurgeon
📞 02-123456
✉️ neurosurgeon@cerebriCare.com

Dr. Jason Stewart
Neurologist
📞 02-987654
✉️ neurologist@cerebriCare.com

Dr. Emily Carter
Clinical Neuropsychologist
📞 02-876543
✉️ neuropsych@cerebriCare.com

CerebriCare - Early Detection for Brain Cancer

127.0.0.1:5000

CerebriCare

Home About Us Doctors News Check Your Condition

Our Doctors

Dr. Thompson Baston
Neurosurgeon
📞 02-123456
✉️ neurosurgeon@cerebriCare.com

Dr. Jason Stewart
Neurologist
📞 02-987654
✉️ neurologist@cerebriCare.com

Dr. Emily Carter
Clinical Neuropsychologist
📞 02-876543
✉️ neuropsych@cerebriCare.com

CerebriCare - Early Detection for Brain Cancer

127.0.0.1:5000

CerebriCare

Home About Us Doctors News Check Your Condition

Latest News

October 01, 2024
Revolutionary Technology for Early Brain Cancer Detection
Discover the latest advancements in technology that enhance early detection of brain cancer.

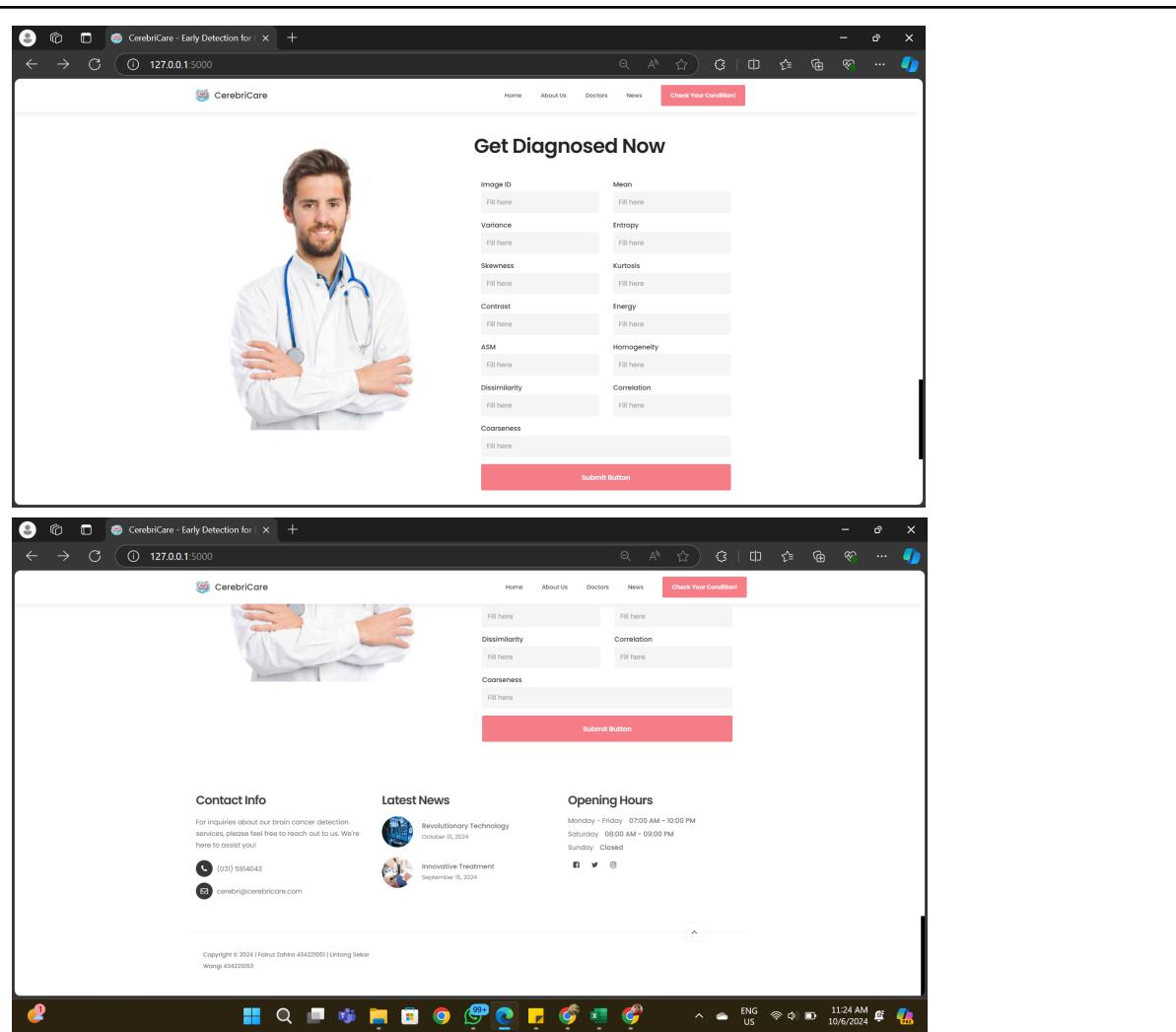
Dr. Sarah Thompson
Neurosurgeon

September 15, 2024
Innovative Treatment Options for Brain Cancer Patients
Discover effective treatments to improve outcomes and quality of life for brain cancer patients.

Dr. John Harris
Neurologist

August 27, 2024
Annual Review of Brain Cancer Research: Key Findings
A comprehensive look at the latest research findings and their implications for treatment.

Dr. Farouz Zahro
Clinical Neuropsychologist

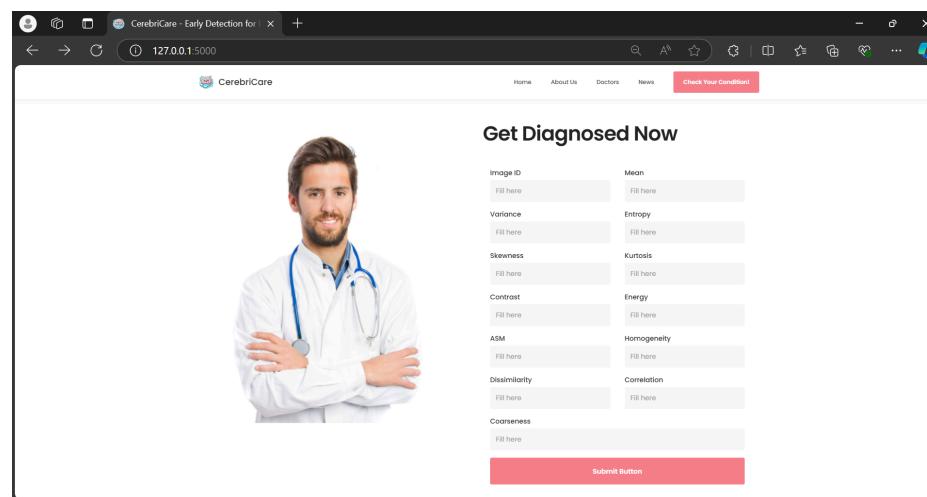


The screenshot shows a web browser window with the URL 127.0.0.1:5000. The page title is "CerebriCare - Early Detection for". The main content area features a doctor's image and a "Get Diagnosed Now" heading. Below this are two rows of input fields for medical analysis metrics: Variance, Skewness, Contrast, ASM, Dissimilarity, Coarseness, Mean, Entropy, Kurtosis, Energy, Homogeneity, and Correlation. A "Submit Button" is at the bottom.

The second part of the screenshot shows the same page after some inputs have been filled. The "Variance" field now contains "Fill here".

Situs CerebriCare adalah platform yang dirancang untuk memberikan layanan diagnosis dini yang mudah diakses dan terpercaya. Dengan integrasi teknologi AI untuk analisis kesehatan otak, situs ini menargetkan pengguna yang mencari solusi inovatif untuk deteksi kanker otak secara efektif dan efisien.

Input Data

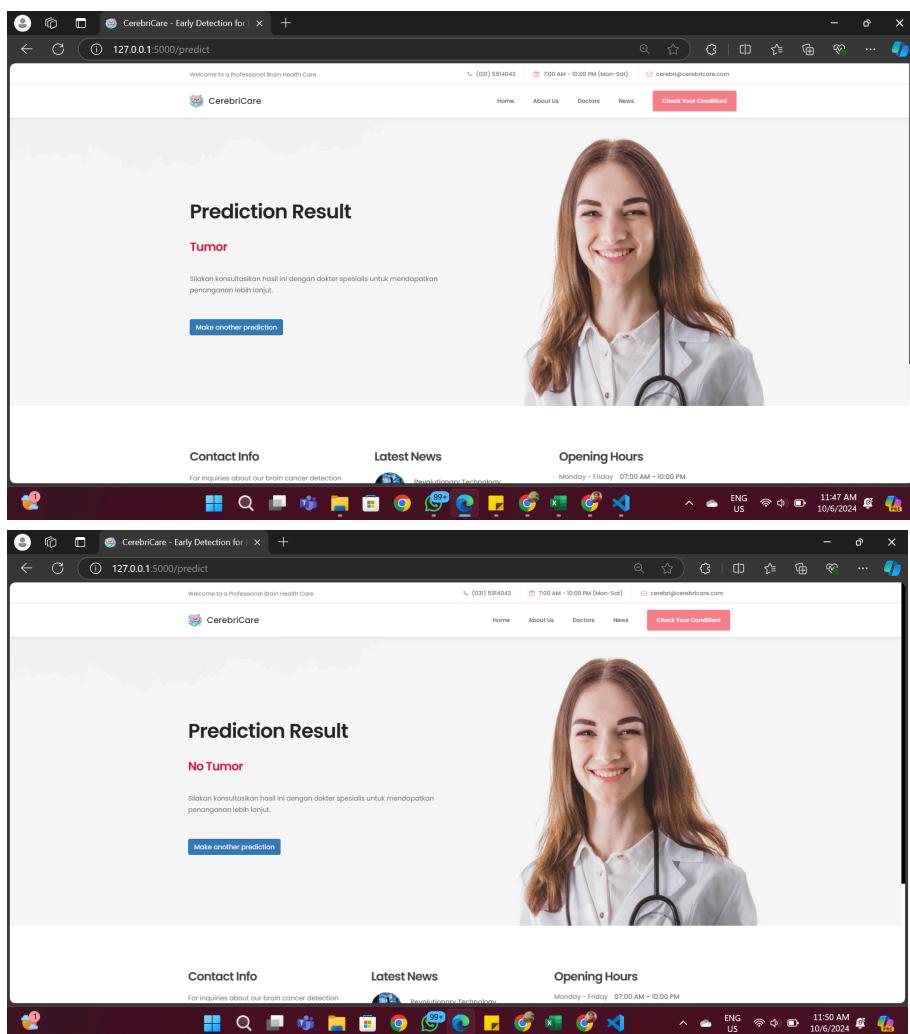


The screenshot shows the "Get Diagnosed Now" form with several input fields filled. The "Variance" field contains "Fill here". The "Mean" field contains "Fill here". Other fields like "Skewness", "Contrast", "ASM", "Dissimilarity", "Coarseness", "Entropy", "Kurtosis", "Energy", "Homogeneity", and "Correlation" also contain "Fill here". A "Submit Button" is at the bottom.

Situs CerebriCare dirancang untuk memberikan layanan diagnosis yang mudah diakses

dan terpercaya. Dengan integrasi teknologi AI untuk analisis kesehatan otak, situs ini menargetkan pengguna yang mencari solusi inovatif untuk deteksi kanker otak secara efektif dan efisien. Pengguna memasukkan hasil analisis gambar MRI otak mereka untuk diproses lebih lanjut, kemudian diberikan prediksi mengenai kemungkinan adanya tumor otak. Model ini menggunakan parameter tekstur dan statistik gambar untuk membuat keputusan diagnostik. Pada dataset Brain Tumor.csv, terdapat kolom-kolom yang sama dengan input di halaman ini, seperti Mean, Variance, Skewness, Entropy, dll. Data-data ini adalah hasil analisis gambar MRI yang sudah dikonversi menjadi nilai numerik dan digunakan sebagai input ke model machine learning. Dengan menginput nilai-nilai ini pada formulir diagnosa, pengguna akan mendapatkan hasil prediksi dari model yang telah dilatih menggunakan dataset tersebut.

Hasil Input Data (Diagnosa)



Kedua halaman ini memberikan informasi prediksi dengan jelas, memastikan bahwa pengguna memahami hasil prediksi. Platform ini juga dirancang untuk mendorong tindakan lebih lanjut dari pengguna, baik hasil prediksi positif maupun negatif, dengan menganjurkan konsultasi ke dokter spesialis. Tombol "Make another prediction" memberikan pengguna kemudahan untuk melakukan proses prediksi tambahan jika diperlukan.

REFERENSI

S. Bauer, R. Wiest, L.-P. Nolte, and M. Reyes, "A survey of MRI-based medical image analysis for brain tumor studies," *Phys. Med. Biol.*, vol. 58, no. 13, pp. R97–R129, 2013.

F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

A.B. Abdusalomov, M. Mukhiddinov, and T.K. Whangbo, "Brain Tumor Detection Based on Deep Learning Approaches and Magnetic Resonance Imaging," *Cancers*, vol. 15, no. 4172, pp. 1–29, 2023.

A. Younis, L. Qiang, C. O. Nyatenga, M. J. Adamu, and H. B. Kawuwa, "Brain Tumor Analysis Using Deep Learning and VGG-16 Ensembling Learning Approaches," *Applied Sciences*, vol. 12, no. 14, pp. 7282, 2022.

F. Taher, M. R. Shoaib, H. M. Emara, K. M. Abdelwahab, F. E. Abd El-Samie, and M. T. Haweel, "Efficient Framework for Brain Tumor Detection Using Different Deep Learning Techniques," *Frontiers in Public Health*, vol. 10, pp. 1–12, 2022.