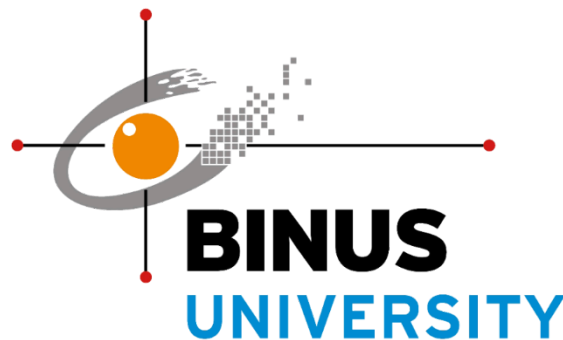


Assignment Report
Earthquake-Prediction
April 2024

Laporan *Data Analytics Assignment*

Oleh

| | |
|-------------------------|------------|
| Saiful Anwar | 2602204361 |
| Muhammda Rizqi Anugerah | 2602187203 |
| Fadhlan Ahmad Radistya | 2602164673 |



Computer Science Program
Computer Science Study Program
School of Computer Science

Universitas Bina Nusantara

Jakarta
2024
Assignment Report
Earthquake-Prediction
April 2024

Laporan *Data Analytics Assignment*

Oleh

| | |
|-------------------------|------------|
| Saiful Anwar | 2602204361 |
| Muhammda Rizqi Anugerah | 2602187203 |
| Fadhlan Ahmad Radistya | 2602164673 |



Computer Science Program
Computer Science Study Program
School of Computer Science

Universitas Bina Nusantara

Jakarta

2024

HALAMAN PERNYATAAN ORISINALITAS

Halaman ini **TIDAK** perlu dibuat, karena akan di-*generate* secara otomatis oleh sistem (website learning plan: <https://enrichment.apps.binus.ac.id/>) ketika Final Report Anda sudah berstatus “***All Approved***” (sudah di *approved* oleh *Site Supervisor* dan *Faculty Supervisor*)

DAFTAR ISI

| | |
|---|-----|
| HALAMAN PERNYATAAN ORISINALITAS | iii |
| DAFTAR ISI | iv |
| DAFTAR GAMBAR | v |
| DAFTAR TABEL | vi |
| BAB I PENDAHULUAN | 1 |
| BAB II STUDI LITERATURE | 11 |
| 2.1 Isi Dataset..... | 3 |
| 2.2 Pemrosesan Sinyal Seismik..... | 4 |
| 2.3 Prediksi Waktu Terjadinya Gempa..... | 5 |
| 2.3.1 Kemampuan Memprediksi Gempa..... | 5 |
| 2.3.2 Random Forest..... | 6 |
| 2.3.3 Support Vector Machine (SVM)..... | 6 |
| 2.3.4 CatBoost..... | 7 |
| 2.4 Analisis Data Eksperimental..... | 8 |
| BAB III METODE | 9 |
| 3.1 Visualisasi Data..... | 9 |
| 3.1.1 Bagian Kode Pertama..... | 10 |
| 3.1.2 Bagian Kode Kedua..... | 10 |
| 3.1.3 Bagian Kode Ketiga..... | 11 |
| 3.2 Feature Engineering..... | 4 |
| BAB IV HASIL IMPLEMENTASI | 9 |
| 4.1 Latih Data..... | 13 |
| 4.1.1 Menyiapkan Data Pelatihan..... | 13 |
| 4.1.2 Hasil Persiapan Data..... | 14 |
| 4.2 Penskalaan Data Latih..... | 14 |

| | |
|--|----|
| 4.3 Uji Coba Model..... | 15 |
| 4.3.1 Random Forest..... | 15 |
| 4.3.2 Support Vector Machine (SVM)..... | 17 |
| 4.3.3 CatBoost..... | 18 |
| 4.3.4 LSTM (Long Short-Term Memory)..... | 20 |
| BAB V KESIMPULAN | 9 |
| 5.1 Proses Data Tes..... | 25 |
| 5.2 Membuat Prediksi..... | 26 |
| REFERENCES | 28 |
| LAMPIRAN | 29 |

DAFTAR GAMBAR

| | |
|--|----|
| Gambar 1. 1 Contoh Data Gempa Bumi | 2 |
| Gambar 2. 1 Data Latih Dengan Nama Train.csv | 3 |
| Gambar 2. 2 Prediksi gempa menggunakan Machine Learning | 5 |
| Gambar 2. 3 Hyperplane yang memisahkan dua kelas positif (+1) dan negatif(-1) | 6 |
| Gambar 2. 4 Algoritma CatBoost | 7 |
| Gambar 2. 5 Tampilan Seismogram Gempa | 8 |
| Gambar 3. 1 Visualisasi Data Ke Grafik/Plot | 9 |
| Gambar 3. 2 Kode Pada Visualisasi | 9 |
| Gambar 3. 3 Potongan Kode 1 | 10 |
| Gambar 3. 4 Potongan Kode 2 | 10 |
| Gambar 3. 5 Potongan Kode 3 | 11 |
| Gambar 3. 6 Kode Pada Feature Engineering | 12 |
| Gambar 4. 1 Kode Penyiapan Latih Data | 13 |
| Gambar 4. 2 Output dari X_train | 14 |
| Gambar 4. 3 Output dari y_train | 14 |
| Gambar 4. 4 Kode Penskalaan | 15 |
| Gambar 4. 5 Model Random Forest | 15 |
| Gambar 4. 6 Hasil Model Random Forest | 16 |
| Gambar 4. 7 Perbandingan Antar Fitur | 16 |
| Gambar 4. 8 Model Support Vector Machine (SVM) | 17 |
| Gambar 4. 9 Hasil Model Support Vector Machine (SVM) | 17 |
| Gambar 4. 10 Model CatBoost | 18 |
| Gambar 4. 11 Hasil Model CatBoost | 19 |
| Gambar 4. 12 Model LSTM (Long Short-Term Memory) | 20 |
| Gambar 4. 13 Model LSTM (Long Short-Term Memory) lanjutan | 23 |
| Gambar 4. 14 Model LSTM (Long Short-Term Memory) lanjutan | 23 |
| Gambar 5. 1 Kode Untuk Proses Data Tes | 25 |
| Gambar 5. 2 Kode Untuk Prediksi Data | 26 |

BAB I

PENDAHULUAN

Gempa bumi merupakan salah satu ancaman global yang serius, mampu menyebabkan kerusakan yang luas dan dampak berkepanjangan terhadap kehidupan manusia serta infrastruktur di seluruh dunia. Fenomena ini tidak mengenal batas geografis, dengan dampak yang dapat dirasakan di berbagai belahan bumi. Sebagai sebuah kejadian alam yang sering kali tidak dapat diprediksi dengan tepat, gempa bumi telah menyebabkan kehancuran dan penderitaan manusia sepanjang sejarah.

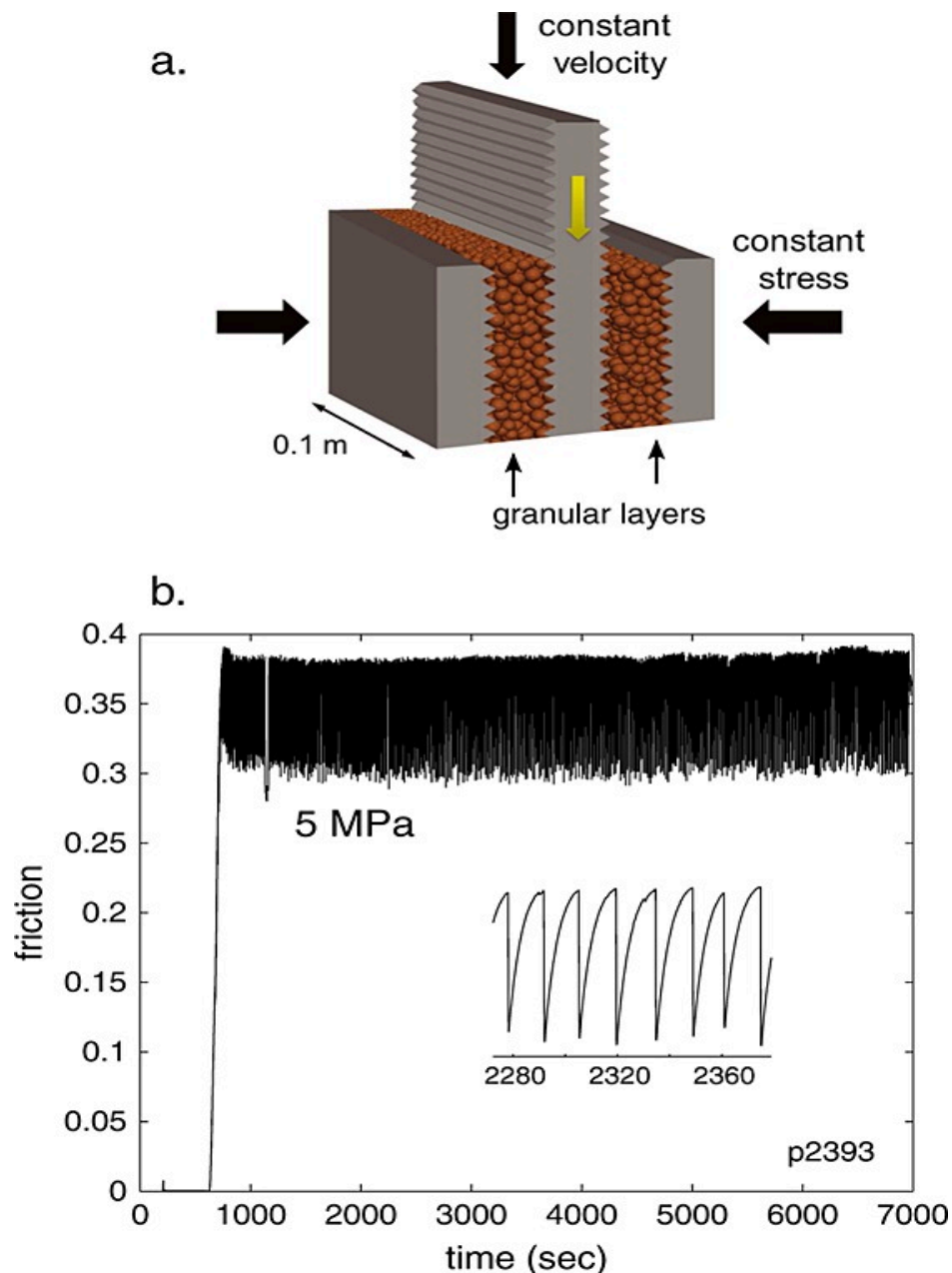
Dengan memperhatikan kompleksitas fenomena ini, serta perlunya langkah-langkah mitigasi yang efektif, perhatian terhadap pengembangan metode prediksi gempa bumi menjadi semakin penting. Melalui upaya-upaya riset dan pengembangan teknologi, para ilmuwan dan ahli telah berusaha mencari cara untuk meningkatkan kemampuan prediksi terkait waktu, lokasi, dan kekuatan gempa bumi di seluruh dunia.

Salah satu pendekatan yang menarik dalam bidang ini adalah penggunaan algoritma Random Forest, sebuah teknik pemodelan yang menggabungkan sejumlah pohon keputusan untuk melakukan prediksi. Kelebihan dari Random Forest termasuk kemampuannya dalam menangani ketergantungan dan interaksi antara variabel yang kompleks, serta mampu mengatasi overfitting. Dengan hasil akurat yang telah terbukti dalam berbagai studi sebelumnya, Random Forest menjadi salah satu metode yang menjanjikan dalam upaya prediksi gempa bumi secara global.

Selain itu, pemodelan prediksi gempa bumi juga memerlukan seleksi fitur yang tepat. Seleksi fitur bertujuan untuk mengidentifikasi variabel-variabel yang paling berpengaruh terhadap terjadinya gempa bumi di seluruh dunia. Dengan memilih fitur-fitur yang relevan dan informatif, diharapkan dapat meningkatkan kualitas prediksi dan mengurangi kompleksitas model.

Berdasarkan tinjauan literatur sebelumnya, terdapat potensi untuk meningkatkan kemampuan prediksi gempa bumi secara global dengan memanfaatkan kombinasi antara Random Forest dan seleksi fitur yang tepat. Melalui penelitian ini, diharapkan dapat memberikan kontribusi yang signifikan dalam upaya mitigasi risiko terhadap gempa bumi di seluruh dunia.

Dengan pemahaman yang lebih baik tentang potensi gempa bumi di masa depan, pihak berwenang dan masyarakat dapat mengambil langkah-langkah mitigasi yang lebih efektif, seperti perencanaan bangunan tahan gempa, evakuasi darurat, dan peningkatan kesadaran akan bahaya gempa bumi. Informasi yang lebih akurat dan dapat diandalkan tentang gempa bumi di seluruh dunia dapat menjadi landasan yang kuat dalam pengambilan keputusan yang lebih baik dalam menghadapi ancaman ini.



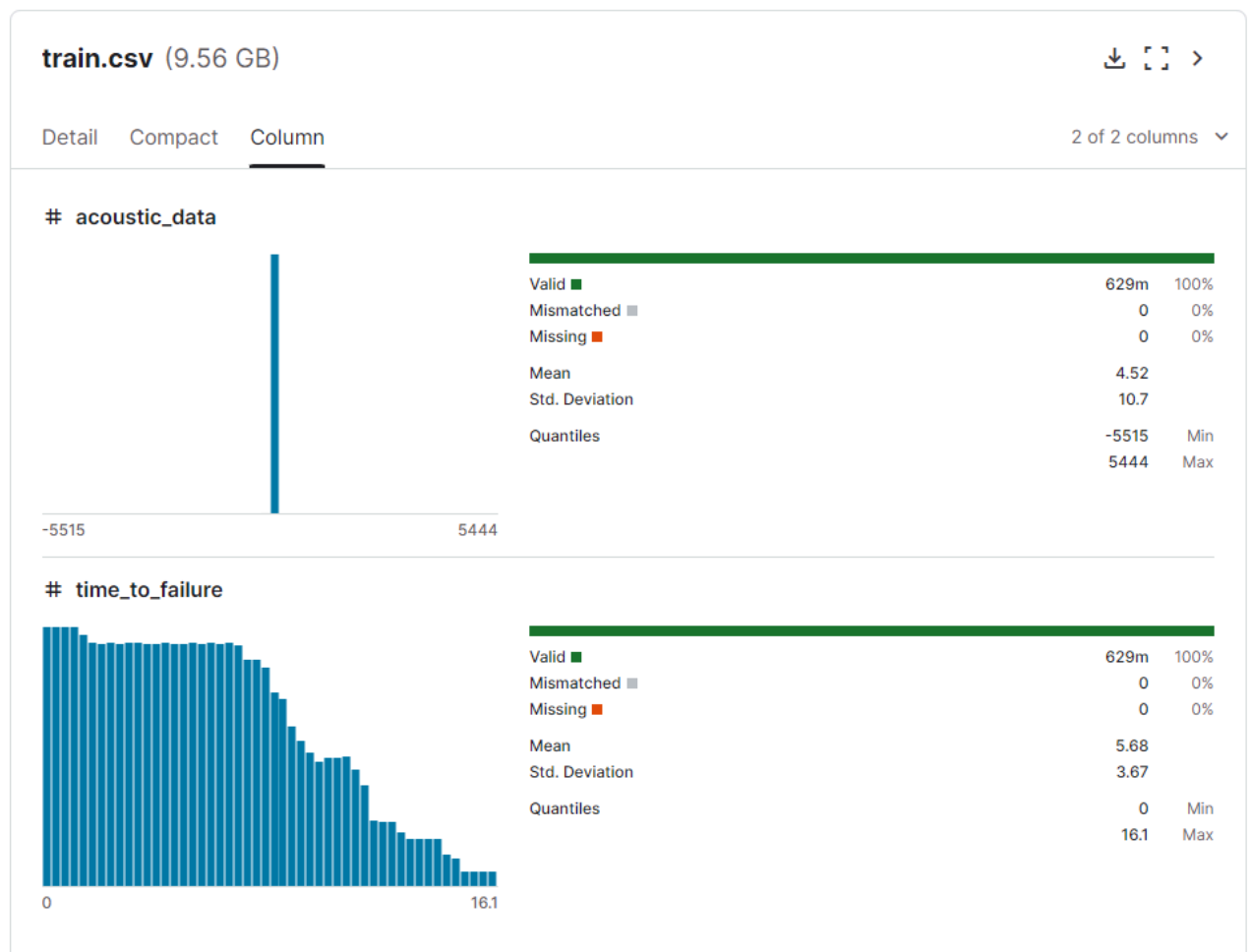
Gambar 1. 1 Contoh Data Gempa Bumi

BAB II

STUDI LITERATURE

2.1 Isi Dataset

Data yang digunakan terdiri dari dua komponen utama: sinyal akustik (`acoustic_data`) dan waktu hingga terjadinya kegagalan (`time_to_failure`). Sinyal akustik merupakan input yang akan digunakan untuk memprediksi waktu yang tersisa sebelum terjadinya gempa bumi laboratorium berikutnya, yang direpresentasikan oleh variabel `time_to_failure`. Data pelatihan berbentuk segmen data eksperimental yang berkelanjutan, sedangkan data pengujian terdiri dari banyak segmen kecil yang terpisah.



Gambar 2. 1 Data Latih Dengan Nama Train.csv

Ringkasan statistik dari dua kolom data pada file `train.csv`, yaitu:

acoustic_data

- Rentang nilai dari -5515 hingga 5444
- Rata-rata (mean) sebesar 4.52
- Standar deviasi 10.7
- Tidak ada data yang hilang (missing) atau tidak cocok (mismatched)

time_to_failure

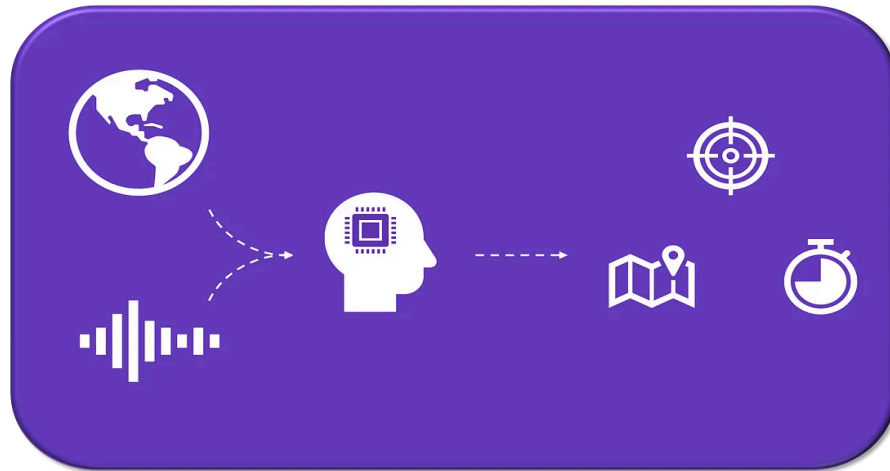
- Rentang nilai dari 0 hingga 16.1
- Rata-rata (mean) sebesar 5.68
- Standar deviasi 3.67
- Tidak ada data yang hilang atau tidak cocok.

2.2 Pemrosesan Sinyal Seismik

Pemrosesan sinyal seismik, sebagai subbidang pemrosesan sinyal digital (DSP), berfokus pada pemrosesan data seismik untuk menekan kebisingan, memperkuat sinyal, dan memindahkan peristiwa seismik ke lokasi yang sesuai di bawah permukaan. Langkah-langkah pemrosesan biasanya mencakup dekonvolusi, analisis kecepatan, pergerakan normal/dip, koreksi statis, penumpukan, dan migrasi. Pemrosesan seismik membantu ahli geologi menerapkan interpretasi yang lebih baik dengan memberikan struktur bawah permukaan yang lebih jelas dan akurat. Tujuan dari prediksi ini adalah menggunakan sinyal seismik untuk memprediksi waktu terjadinya gempa. [1]

2.3 Prediksi Waktu Terjadinya Gempa

2.3.1 Kemampuan Memprediksi Gempa



Gambar 2. 2 Prediksi gempa menggunakan Machine Learning

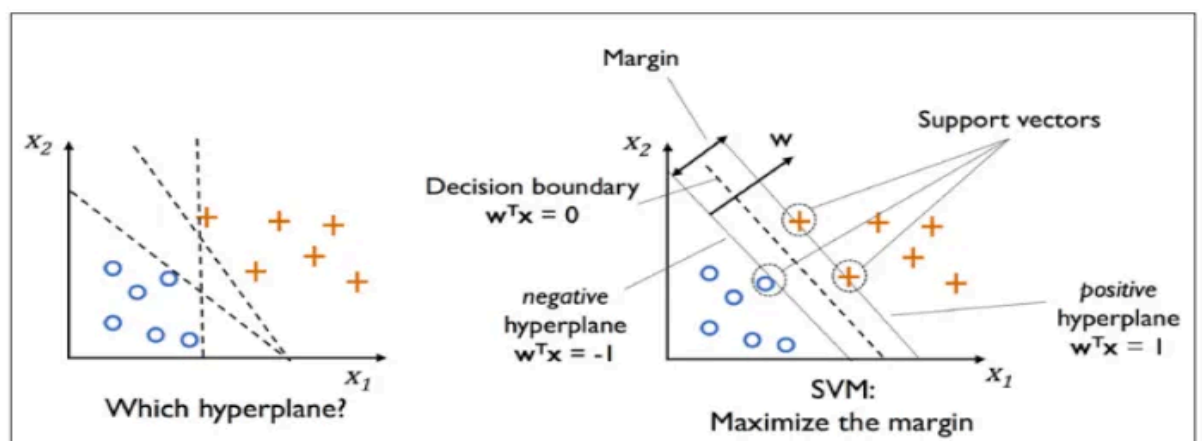
Prediksi gempa bumi merupakan bidang yang kompleks dan sulit diprediksi secara tepat. Meskipun penelitian dan teknologi terus berkembang, saat ini tidak ada metode yang dapat memberikan prediksi yang akurat dalam menentukan waktu, lokasi, dan kekuatan gempa bumi di suatu wilayah secara spesifik. Prediksi kapan terjadinya gempa bumi di suatu wilayah perlu dilakukan untuk mengurangi jumlah korban jiwa. Salah satu metode yang dapat digunakan untuk prediksi gempa bumi adalah Random Forest. Random Forest adalah sebuah teknik pemodelan yang menggabungkan sejumlah pohon keputusan (decision trees) untuk melakukan prediksi. Kelebihan dari Random Forest adalah kemampuannya dalam menangani ketergantungan dan interaksi antara variabel yang kompleks serta mampu mengatasi overfitting. Selain itu, Random Forest dapat menghasilkan kesalahan yang cukup rendah, kinerja yang optimal dalam klasifikasi, mampu menangani data pelatihan dalam jumlah besar dengan efisien, serta metode yang efektif untuk memperkirakan data yang hilang. [2]

2.3.2 Random Forest

Random Forest adalah sebuah teknik pemodelan yang menggabungkan sejumlah pohon keputusan (decision trees) untuk melakukan prediksi. Kelebihan dari Random Forest adalah kemampuannya dalam menangani ketergantungan dan interaksi antara variabel yang kompleks serta mampu mengatasi overfitting. Selain itu, Random Forest dapat menghasilkan kesalahan yang cukup rendah, kinerja yang optimal dalam klasifikasi, mampu menangani data pelatihan dalam jumlah besar dengan efisien, serta metode yang efektif untuk memperkirakan data yang hilang. [2]

2.3.3 Support Vector Machine (SVM)

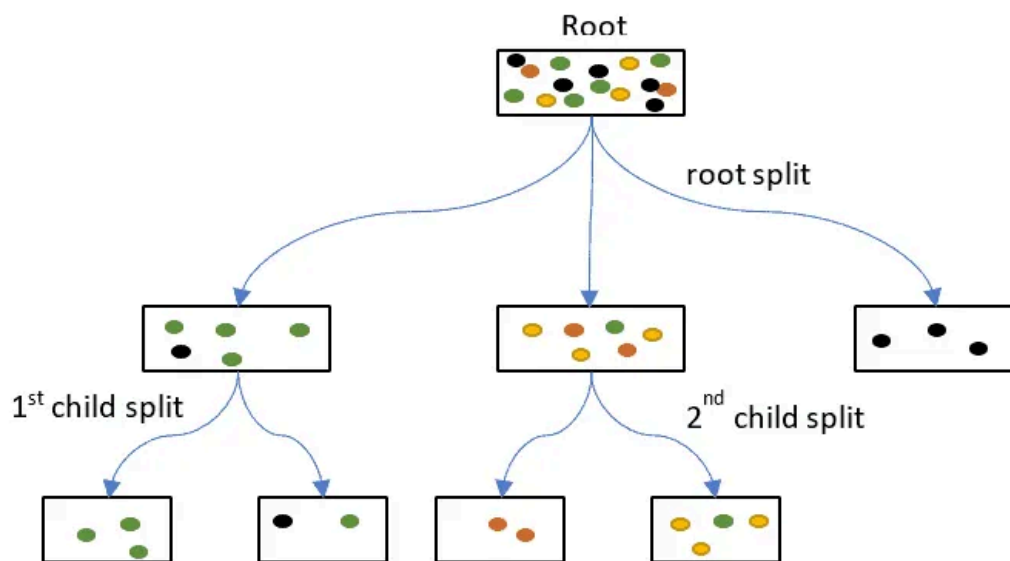
Support Vector Machine (SVM) merupakan salah satu metode dalam supervised learning yang biasanya digunakan untuk klasifikasi (seperti Support Vector Classification) dan regresi (Support Vector Regression). Dalam pemodelan klasifikasi, SVM memiliki konsep yang lebih matang dan lebih jelas secara matematis. SVM digunakan untuk mencari hyperplane terbaik dengan memaksimalkan jarak antar kelas. Hyperplane adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas. Dalam 2-D fungsi yang digunakan untuk klasifikasi antar kelas disebut sebagai line whereas, fungsi yang digunakan untuk klasifikasi antar kelas dalam 3-D disebut plane similarly, sedangkan fungsi yang digunakan untuk klasifikasi di dalam ruang kelas dimensi yang lebih tinggi di sebut hyperplane. [3]



Gambar 2. 3 Hyperplane yang memisahkan dua kelas positif (+1) dan negatif(-1)

2.3.4 CatBoost

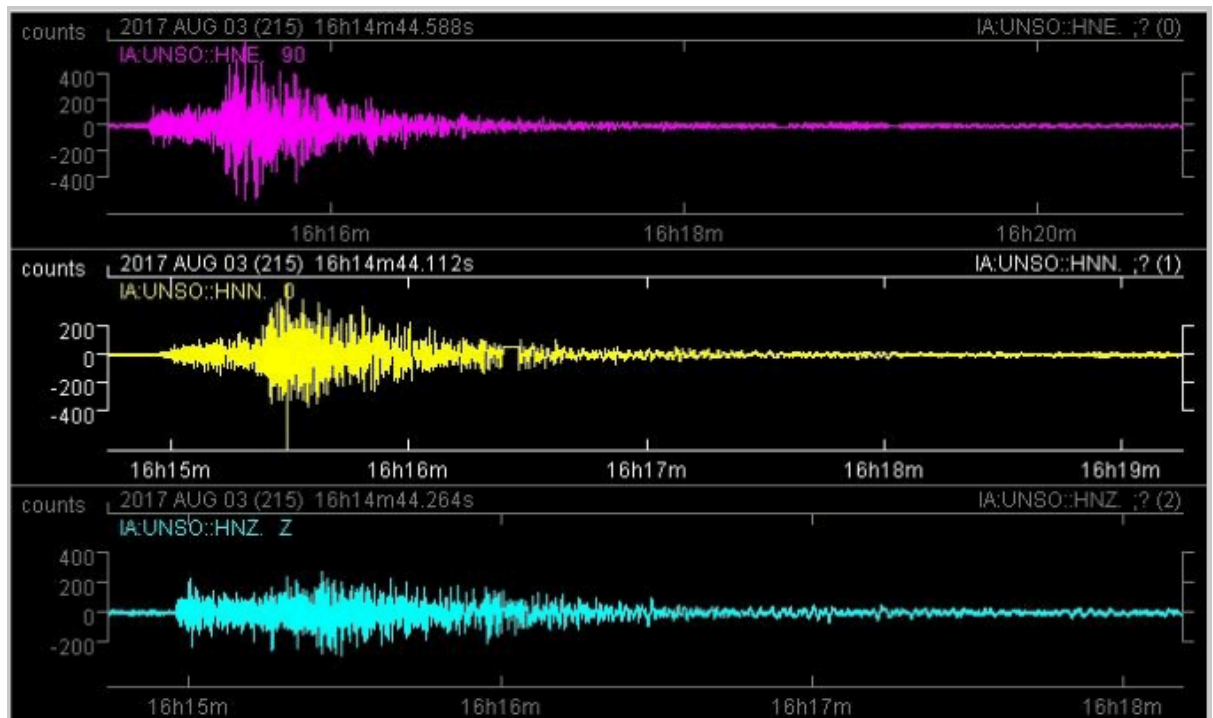
CatBoost merupakan singkatan dari “Category Boosting,” yang menunjukkan keunggulan algoritma ini dalam menangani fitur kategorikal dalam data. Algoritma CatBoost dikembangkan untuk meningkatkan performa model Machine Learning dengan fokus pada kecepatan, akurasi, dan kemampuan penanganan fitur kategorikal. CatBoost didasarkan pada algoritma Gradient Boosting yang telah terbukti sukses, namun dengan adanya inovasi dan penyesuaian tertentu. Pengembang CatBoost memperkenalkan beberapa fitur dan teknik unik yang membedakannya dari algoritma Boosting lainnya, termasuk penanganan otomatis terhadap fitur kategorikal, penanganan missing values, dan fitur “Ordered Boosting” yang membantu mengatasi overfitting. [4]



Gambar 2. 4 Algoritma CatBoost

2.4 Analisis Data Eksperimental

Dalam konteks ini, prediksi waktu terjadinya gempa bumi merupakan tahapan kritis dalam upaya memahami pola dan dinamika seismik yang terkait. Langkah awal dalam analisis ini adalah memanfaatkan data seismik yang sudah ada. Data yang diberikan berupa satu segmen data eksperimental yang kontinu, yang menyediakan isi terhadap perilaku seismik dalam skenario laboratorium. Selanjutnya, teknik-teknik analisis data digunakan untuk mengolah data ini, dengan fokus pada ekstraksi fitur-fitur yang relevan untuk memahami dan meramalkan waktu terjadinya gempa. Proses ekstraksi fitur melibatkan pembentukan instance pelatihan dengan menggunakan 150.000 sekuens data, yang kemudian digunakan untuk menghasilkan berbagai fitur statistik yang mewakili karakteristik sinyal seismik.

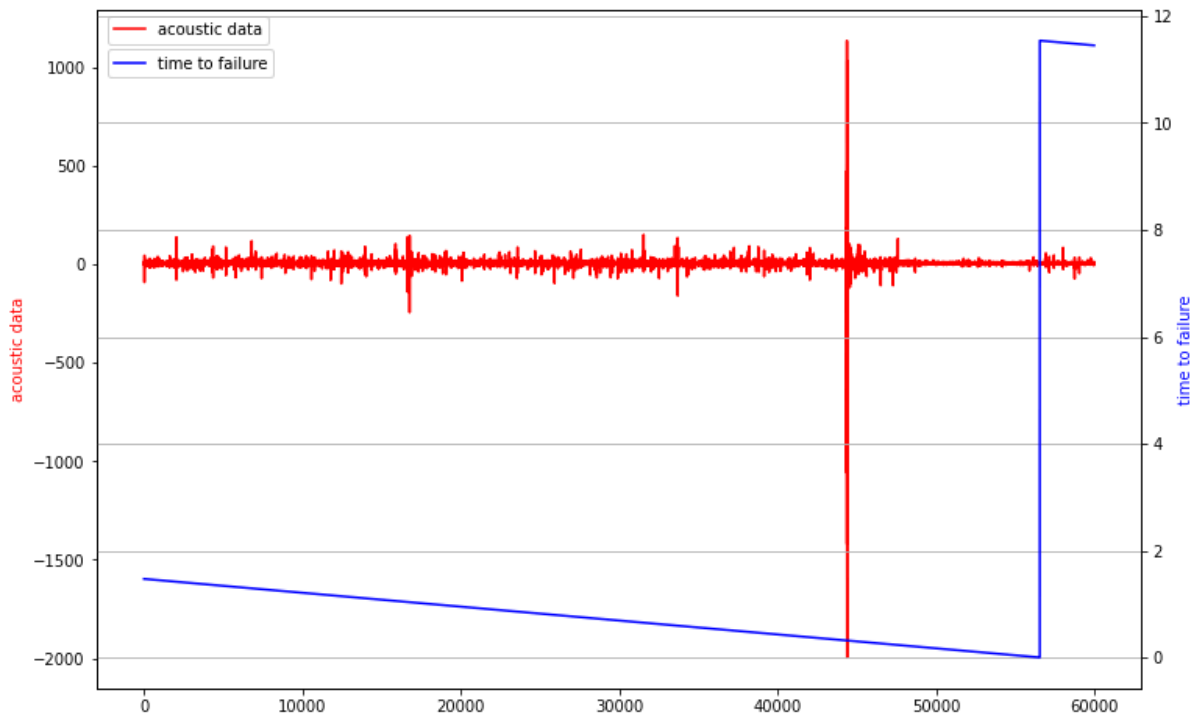


Gambar 2. 5 Tampilan Seismogram Gempa

BAB III

METODE

3.1 Visualisasi Data



Gambar 3. 1 Visualisasi Data Ke Grafik/Plot

```
train_ad_sample = train_df['acoustic_data'].values[::100] # :: means skip by 100
train_ttf_sample = train_df['time_to_failure'].values[::100]

fig, ax1 = plt.subplots(figsize=(12, 8))

plt.plot(train_ad_sample, color='r')
ax1.set_ylabel('acoustic data', color='r')
plt.legend(['acoustic data'], loc=(0.01, 0.95))

ax2 = ax1.twinx()

plt.plot(train_ttf_sample, color='b')
ax2.set_ylabel('time to failure', color='b')
plt.legend(['time to failure'], loc=(0.01, 0.9))

plt.grid(True)
```

Gambar 3. 2 Kode Pada Visualisasi

3.1.1 Bagian Kode Pertama

```
train_ad_sample = train_df['acoustic_data'].values[::100] # :: means skip by 100
train_ttf_sample = train_df['time_to_failure'].values[::100]
```

Gambar 3. 3 Potongan Kode 1

Pada bagian ini, kami melakukan sampling pada data pelatihan **train_df** dengan mengambil setiap 100 baris data dari kolom **'acoustic_data'** dan **'time_to_failure'**. Sampling ini bertujuan untuk mengurangi kepadatan data yang akan divisualisasikan, sehingga plot akan lebih mudah dibaca dan dianalisis.

3.1.2 Bagian Kode Kedua

```
fig, ax1 = plt.subplots(figsize=(12, 8))

plt.plot(train_ad_sample, color='r')
ax1.set_ylabel('acoustic data', color='r')
plt.legend(['acoustic data'], loc=(0.01, 0.95))
```

Gambar 3. 4 Potongan Kode 2

Membuat sebuah figure baru menggunakan **plt.subplots()**. Kemudian, data **train_ad_sample** (acoustic data yang diambil sampelnya) diplot dengan garis berwarna merah menggunakan **plt.plot()**. Label sumbu y untuk acoustic data diatur dengan **ax1.set_ylabel()**, dan legenda untuk plot acoustic data ditambahkan dengan **plt.legend()**.

3.1.3 Bagian Kode Ketiga

```
ax2 = ax1.twinx()

plt.plot(train_ttf_sample, color='b')
ax2.set_ylabel('time to failure', color='b')
plt.legend(['time to failure'], loc=(0.01, 0.9))

plt.grid(True)
```

Gambar 3. 5 Potongan Kode 3

Kemudian, sumbu y kedua (**ax2**) dibuat dengan **ax1.twinx()**, yang memungkinkan ada dua sumbu y dengan skala yang berbeda dalam satu plot. Data **train_ttf_sample** (time to failure yang diambil sampelnya) kemudian diplot dengan garis berwarna biru menggunakan **plt.plot()**.

Alasan menggunakan visualisasi seperti *Gambar 3.1* adalah untuk memudahkan perbandingan dan analisis hubungan antara dua variabel penting, yaitu **acoustic_data** dan **time_to_failure**. Dengan menampilkan keduanya dalam satu plot, dapat lebih mudah melihat bagaimana perubahan pada sinyal akustik (**acoustic_data**) berkaitan dengan waktu hingga kegagalan (**time_to_failure**).

3.2 Feature Engineering

Feature engineering adalah proses memanipulasi variabel mentah dari sebuah dataset untuk membuat fitur-fitur baru yang lebih berguna dan memberikan informasi lebih baik untuk tugas analisis atau pemodelan machine learning tertentu [5].

```
def gen_features(X):  
    strain = []  
    strain.append(X.std())  
    strain.append(X.min())  
    strain.append(X.max())  
    strain.append(np.abs(X).max())  
    strain.append(np.abs(X).std())  
    return pd.Series(strain)
```

Gambar 3. 6 Kode Pada Feature Engineering

Fungsi **gen_features** mengambil sebuah array data X dan menghasilkan:

- ☐ Standar deviasi (**X.std()**)
- ☐ Nilai minimum (**X.min()**)
- ☐ Nilai maksimum (**X.max()**)
- ☐ Nilai maksimum dari nilai absolut (**np.abs(X).max()**)
- ☐ Standar deviasi dari nilai absolut (**np.abs(X).std()**)

Menurut kami, proses Feature Engineering ini penting karena model pembelajaran akan bekerja lebih baik karena dengan membuat fitur-fitur baru dari data mentah, kami dapat meningkatkan kinerja model.

BAB IV

HASIL IMPLEMENTASI

4.1 Latih Data

4.1.1 Menyiapkan Data Pelatihan

```
train = pd.read_csv('train.csv', iterator = True, chunksize = 150000,
dtype = {'acoustic_data': np.int16, 'time_to_failure': np.float64})
X_train = pd.DataFrame()
y_train = pd.Series()

for df in train:
    ch = gen_features(df['acoustic_data'])
    X_train = X_train.append(ch, ignore_index=True)
    y_train = y_train.append(pd.Series(df['time_to_failure'].values[-1]))
```

Gambar 4. 1 Kode Penyiapan Latih Data

Tujuan kami adalah untuk membaca data dari file CSV '**train.csv**' secara bertahap (chunk-by-chunk) menggunakan pandas. Ini sangat berguna karena data yang akan dibaca terlalu besar untuk dimuat sekaligus ke dalam memori, sehingga dibaca secara bertahap dalam potongan-potongan (chunk).

- Tahap awal, code akan membaca file '**train.csv**' secara bertahap dengan ukuran chunk sebesar 150.000 baris pada bagian **train = pd.read_csv('train.csv', iterator=True, chunksize=150000, dtype={'acoustic_data': np.int16, 'time_to_failure': np.float64})**.
- Dtype harus kami tentukan untuk mengoptimalkan penggunaan memori, karena kolom '**acoustic_data**' dianggap sebagai **np.int16 (16-bit integer)** dan kolom '**time_to_failure**' dianggap sebagai **np.float64 (float)**.
- Kemudian, membuat sebuah dataframe kosong yang akan digunakan untuk menyimpan fitur-fitur dari data pelatihan dengan **X_train = pd.DataFrame()**.
- **y_train = pd.Series()**: berfungsi sebagai frame kosong yang akan digunakan untuk menyimpan label (waktu tersisa hingga kegagalan) dari data pelatihan.

Dengan cara ini, kode diharapkan mampu membaca data secara bertahap dari file CSV, menghasilkan fitur-fitur dan label untuk setiap chunk data, dan menyimpannya ke dalam dataframe `X_train` dan series `y_train` secara bertahap.

4.1.2 Hasil Persiapan Data

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|-------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|
| count | 4195.000000 | 4195.000000 | 4195.000000 | 4195.000000 | 4195.000000 | 4195.000000 | 4195.000000 |
| mean | 6.547788 | -149.190942 | 163.522288 | 68.297997 | 0.125830 | 170.046246 | 5.750165 |
| std | 8.503939 | 265.087984 | 272.930331 | 70.532565 | 0.477901 | 296.887015 | 8.339211 |
| min | 2.802720 | -5515.000000 | 23.000000 | 0.648602 | -4.091826 | 23.000000 | 2.589085 |
| 25% | 4.478637 | -154.000000 | 92.000000 | 28.090227 | -0.040779 | 94.000000 | 3.862810 |
| 50% | 5.618798 | -111.000000 | 123.000000 | 45.816625 | 0.085620 | 127.000000 | 4.781513 |
| 75% | 6.880904 | -79.000000 | 170.000000 | 78.664202 | 0.253930 | 175.000000 | 5.887947 |
| max | 153.703569 | -15.000000 | 5444.000000 | 631.158927 | 4.219429 | 5515.000000 | 150.432368 |

Gambar 4. 2 Output dari `X_train`

```
count    4195.000000
mean      5.683670
std       3.673246
min       0.006398
25%       2.635348
50%       5.358796
75%       8.177500
max      16.103196
dtype: float64
```

Gambar 4. 3 Output dari `y_train`

4.2 Penskalaan Data Latih

proses scaling pada fitur-fitur yang dihasilkan dari data pelatihan menggunakan metode `StandardScaler` dari modul preprocessing dalam library `scikit-learn`. Proses scaling bertujuan untuk mengubah skala setiap fitur sehingga memiliki mean nol dan variansi satu. Dengan begitu, fitur-fitur dalam data pelatihan diubah sedemikian rupa sehingga memiliki skala yang seragam dan memudahkan dalam proses pembelajaran oleh model machine learning.

```
# scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
```

Gambar 4. 4 Kode Penskalaan

- **scaler = StandardScaler()**: yang akan digunakan untuk melakukan scaling pada data.
- Metode **fit()** dipanggil pada objek scaler **scaler.fit(X_train)**, yang menghitung mean dan standard deviation dari setiap fitur dalam data pelatihan **X_train**. Hal ini akan digunakan nanti untuk melakukan scaling.
- **X_train_scaled = scaler.transform(X_train)**: Pada **transform()** dipanggil pada objek scaler untuk melakukan scaling pada data pelatihan **X_train** berdasarkan mean dan standard deviation yang telah dihitung sebelumnya. Hasilnya, **X_train_scaled**, adalah data yang telah diubah skala dengan mean nol dan variansi satu.

4.3 Uji Coba Model

4.3.1 Random Forest

```
#random forest

rand_forest = RandomForestRegressor(n_estimators=100, criterion='absolute_error', oob_score=True, n_jobs=-1)
start_time = time.time()
rand_forest.fit(X_train_scaled, y_train.values.flatten())
print("Training Time:", time.time()-start_time)
print('Training Score:', rand_forest.score(X_train_scaled, y_train.values.flatten()))
rand_forest.oob_score_
```

Gambar 4. 5 Model Random Forest

Membuat objek model Random Forest Regressor dengan menerapkan parameter-parameter berikut:

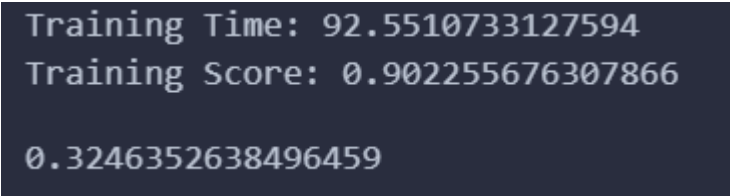
n_estimators=100: Menentukan jumlah pohon keputusan yang akan digunakan dalam ensemble.

criterion='absolute_error': Menggunakan kriteria "absolute error" untuk mengukur kualitas split. Karena kami akan mencoba untuk meminimalkan nilai absolut dari selisih antara nilai aktual dan nilai prediksi.

oob_score=True: Menghitung out-of-bag (OOB) score, yang merupakan metrik evaluasi untuk mengukur performa model menggunakan sampel yang tidak digunakan dalam proses bootstrap.

rand_forest.fit(X_train_scaled, y_train.values.flatten()): Melatih model Random Forest menggunakan fitur-fitur yang telah diubah skala (X_train_scaled) dan label (y_train).

rand_forest.oob_score_: Menampilkan nilai out-of-bag (OOB) score dari model, yang memberikan indikasi tentang kinerja model pada data yang tidak digunakan dalam proses bootstrap.



```
Training Time: 92.5510733127594
Training Score: 0.902255676307866
0.3246352638496459
```

Gambar 4. 6 Hasil Model Random Forest

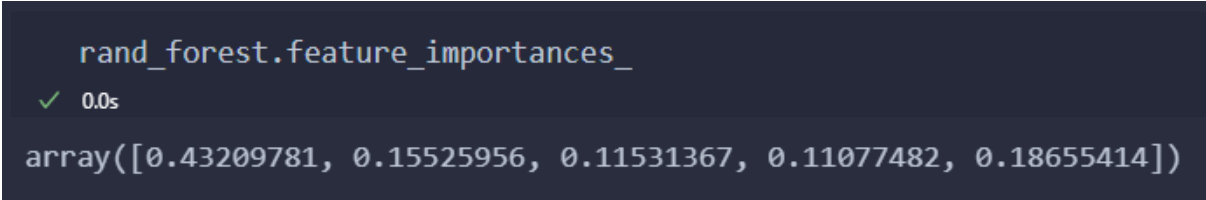
Diberikan hasil sebagai berikut:

Waktu pelatihan model adalah sekitar 92.55 detik.

Skor pelatihan model adalah sekitar 0.9023.

OOB score dari model adalah sekitar 0.3246.

Ini memberikan gambaran awal tentang performa dan kecocokan model Random Forest terhadap data pelatihan.



```
rand_forest.feature_importances_
✓ 0.0s
array([0.43209781, 0.15525956, 0.11531367, 0.11077482, 0.18655414])
```

Gambar 4. 7 Perbandingan Antar Fitur

1. Fitur pertama memiliki tingkat efektifan sebesar 0.4321.
2. Fitur kedua memiliki tingkat efektifan sebesar 0.1553.
3. Fitur ketiga memiliki tingkat efektifan sebesar 0.1153.
4. Fitur keempat memiliki tingkat efektifan sebesar 0.1108.
5. Fitur kelima memiliki tingkat efektifan sebesar 0.1866.

Tingkat penting ini berpengaruh terhadap seberapa besar ke efektifan masing-masing fitur dalam memprediksi. Semakin tinggi nilai tingkat penting, semakin besar kontribusi fitur tersebut terhadap kemampuan model untuk melakukan prediksi.

4.3.2 Support Vector Machine (SVM)

```
#svm

svr = SVR(kernel = 'rbf', tol = 0.01, C= 2, gamma=0.02)
start_time = time.time()
svr.fit(X_train_scaled, y_train.values.flatten())
print("Training Time:", time.time()-start_time)
print('Training Score:', svr.score(X_train_scaled, y_train.values.flatten()))
```

✓ 1.1s

Gambar 4. 8 Model Support Vector Machine (SVM)

kernel='rbf': Menggunakan kernel radial basis function (RBF) untuk mengubah ruang fitur menjadi dimensi yang lebih tinggi, memungkinkan pemisahan yang lebih baik antara kelas..

gamma=0.02: Menentukan koefisien kernel untuk kernel 'rbf'.

start_time = time.time(): Waktu awal untuk mengukur berapa lama pelatihan model.

svr.fit(X_train_scaled, y_train.values.flatten()): Melatih model SVR menggunakan fitur-fitur yang telah diubah skala (X_train_scaled) dan label (y_train).

```
Training Time: 0.5616121292114258
Training Score: 0.3044303767636417
```

Gambar 4. 9 Hasil Model Support Vector Machine (SVM)

Diberikan:

Waktu pelatihan model adalah sekitar 0.5616 detik.

Skor pelatihan model adalah sekitar 0.3044.

Tampaknya model Random Forest memiliki skor pelatihan yang lebih tinggi (sekitar 0.9023) dibandingkan dengan skor pelatihan model SVR (sekitar 0.3044). Secara umum, semakin tinggi skor pelatihan, semakin baik model dapat mempelajari pola-pola yang ada dalam data pelatihan. Model Random Forest kemungkinan lebih baik dalam mempelajari pola-pola yang ada dalam data pelatihan dibandingkan dengan model SVR.

4.3.3 CatBoost

```
#catboost

pool = Pool(X_train, y_train)
cat_boost = CatBoostRegressor(loss_function='MAE', boosting_type='Ordered') # iterations=no. of trees
start_time = time.time()
cat_boost.fit(pool, silent=True) # don't show training | verbose
print("Time taken to train:", time.time()-start_time)
cat_boost.best_score_

✓ 4.7s
```

Gambar 4. 10 Model CatBoost

pool = Pool(X_train, y_train): Di awal, membuat objek Pool dari data pelatihan dan labelnya. Pool adalah struktur data khusus yang digunakan oleh CatBoost untuk menyimpan data pelatihan dan labelnya.

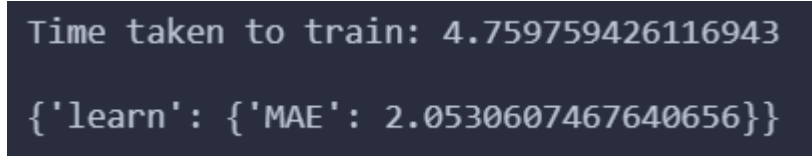
cat_boost = CatBoostRegressor(loss_function='MAE', boosting_type='Ordered'): Membuat objek model CatBoostRegressor dengan menerapkan parameter-parameter berikut:

loss_function='MAE': Dengan menetapkan fungsi kerugian (loss function) sebagai 'MAE' (Mean Absolute Error). Model akan menggunakan MAE sebagai metrik evaluasi utama.

boosting_type='Ordered': Salah satu jenis metode boosting yang digunakan oleh CatBoost.

cat_boost.fit(pool, silent=True): Melatih model CatBoost menggunakan data yang telah dipersiapkan sebelumnya dalam objek pool. Parameter `silent=True` digunakan untuk menyembunyikan output pelatihan dan menjaga konsol tetap bersih.

cat_boost.best_score_: Menampilkan skor terbaik yang diperoleh selama pelatihan model. Dalam hal ini, MAE terbaik yang diperoleh selama pelatihan adalah sekitar 2.0531.



```
Time taken to train: 4.759759426116943
{'learn': {'MAE': 2.0530607467640656}}
```

Gambar 4. 11 Hasil Model CatBoost

- Waktu pelatihan model adalah sekitar 4.7597 detik.
- MAE terbaik yang diperoleh selama pelatihan adalah sekitar 2.0531.

Bisa dibilang, model CatBoost memberikan hasil MAE yang cukup baik, dan waktu pelatihan yang relatif cepat.

Jika hanya mempertimbangkan performa pelatihan yang diamati sejauh ini, model Random Forest memiliki skor pelatihan yang lebih tinggi dibandingkan dengan model SVR dan CatBoost. Namun, performa pelatihan tidak selalu mencerminkan performa pada data uji yang sebenarnya.

4.3.4 LSTM (Long Short-Term Memory)

LSTM adalah jenis arsitektur recurrent neural network (RNN) yang sering digunakan untuk menangani data urutan, seperti teks dan waktunya series data.

```
rnn = Sequential()

rnn.add(LSTM(2, input_shape=X_train_seq.shape[1:], return_sequences=True))
rnn.add(Dropout(rate=0.2))
rnn.add(BatchNormalization())

#rnn.add(LSTM(150, input_shape=X_train_seq.shape[1:], return_sequences=True))
#rnn.add(Dropout(rate=0.2))
#rnn.add(BatchNormalization())

rnn.add(LSTM(2, input_shape=X_train_seq.shape[1:]))
rnn.add(Dropout(rate=0.2))
rnn.add(BatchNormalization())

rnn.add(Dense(1, activation='elu'))
rnn.add(Dropout(rate=0.2))

rnn.add(Dense(1))
```

Gambar 4. 12 Model LSTM (Long Short-Term Memory)

Model ini menggunakan arsitektur dengan beberapa lapisan LSTM, Dropout, BatchNormalization, dan lapisan Dense.

rnn.add(LSTM(2, input_shape=X_train_seq.shape[1:], return_sequences=True)): Menambahkan lapisan LSTM pertama ke dalam model. Lapisan ini memiliki 2 unit LSTM, dengan input shape yang sesuai dengan dimensi dari setiap sampel fitur dalam data pelatihan (**X_train_seq.shape[1:]**).

return_sequences=True menandakan bahwa lapisan ini menghasilkan output untuk setiap langkah waktu dalam urutan.

rnn.add(Dropout(rate=0.2)): Menambahkan lapisan Dropout pertama ke dalam model. Lapisan ini digunakan untuk mengurangi overfitting secara acak yang dapat menonaktifkan sebagian unit selama pelatihan, dengan tingkat dropout sebesar 0.2 (20%).

rnn.add(BatchNormalization()): Menambahkan lapisan BatchNormalization pertama ke dalam model. Lapisan ini digunakan untuk mengnormalisasi input dari lapisan sebelumnya.

rnn.add(LSTM(2, input_shape=X_train_seq.shape[1:])): Menambahkan lapisan LSTM kedua ke dalam model. Lapisan ini memiliki 2 unit LSTM, dan input shape yang sesuai dengan output dari lapisan sebelumnya. Karena tidak ada **return_sequences=True**, lapisan ini hanya menghasilkan output untuk langkah waktu terakhir dalam urutan.

rnn.add(Dropout(rate=0.2)): Menambahkan lapisan Dropout kedua ke dalam model, dengan tingkat dropout sebesar 0.2.

rnn.add(BatchNormalization()): Menambahkan lapisan BatchNormalization kedua ke dalam model.

rnn.add(Dense(1, activation='elu')): Menambahkan lapisan Dense ke dalam model dengan 1 unit dan fungsi aktivasi 'elu'.

rnn.add(Dropout(rate=0.2)): Menambahkan lapisan Dropout ketiga ke dalam model, dengan tingkat dropout sebesar 0.2.

rnn.add(Dense(1)): Menambahkan lapisan Dense (output) ke dalam model dengan 1 unit. Ini adalah lapisan output yang menghasilkan prediksi waktu-to-failure.

```

opt = tf.keras.optimizers.Adam(learning_rate=0.001, decay=1e-6)
rnn.compile(loss='mean_absolute_error', optimizer=opt, metrics=['mae'])
✓ 0.0s

rnn.summary()
✓ 0.0s
Model: "sequential"

```

| Layer (type) | Output Shape | Param # |
|---|-------------------|---------|
| lstm (LSTM) | (None, 150000, 2) | 32 |
| dropout (Dropout) | (None, 150000, 2) | 0 |
| batch_normalization (Batch Normalization) | (None, 150000, 2) | 8 |
| lstm_1 (LSTM) | (None, 2) | 40 |
| dropout_1 (Dropout) | (None, 2) | 0 |
| batch_normalization_1 (Batch Normalization) | (None, 2) | 8 |
| dense (Dense) | (None, 1) | 3 |
| dropout_2 (Dropout) | (None, 1) | 0 |
| dense_1 (Dense) | (None, 1) | 2 |

```

Total params: 93
Trainable params: 85
Non-trainable params: 8

```

Gambar 4. 13 Model LSTM (Long Short-Term Memory) lanjutan

opt = tf.keras.optimizers.Adam(learning_rate=0.001, decay=1e-6): Kami mendefinisikan optimizer Adam dengan laju pembelajaran (learning rate) 0.001 dan nilai decay 1e-6. Optimizer Adam adalah algoritma optimisasi yang efisien dan sering digunakan dalam pelatihan model neural network.

rnn.compile(loss='mean_absolute_error', optimizer=opt, metrics=['mae']):

Mengompilasi model dengan menggunakan loss function Mean Absolute Error

(MAE), optimizer yang telah didefinisikan sebelumnya (opt), dan menambahkan metrik evaluasi MAE untuk mengevaluasi kinerja model saat pelatihan.

rnn.summary(): Ringkasan (summary) dari model yang telah dibuat.

Dapat melihat arsitektur model secara keseluruhan, termasuk lapisan-lapisan yang ditambahkan sebelumnya, beserta jumlah parameter yang dapat dilatih dan non-trainable. Bisa dilihat, jumlah total parameter adalah 93, di mana 85 parameter dapat dilatih dan 8 parameter tidak dapat dilatih (non-trainable). Ini karena lapisan BatchNormalization memiliki parameter yang tidak dapat dilatih.

```
time1 = time.time()
rnn.fit(X_train_seq, y_train_seq, epochs=2, batch_size=4)
print("Training Time:", time.time()-time1)

Epoch 1/2
2024-04-03 21:49:46.846109: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 12000000 exceeds 10% of free system memory.

X_test_seq = []

for file in os.listdir('test'):
    data = pd.read_csv(os.path.join('test', file), dtype = {'acoustic_data': np.int16})
    X_test_seq.append(data['acoustic_data'].to_numpy().reshape(-1,1))

# This takes more than 45 min
#X_pred_rnn = rnn.predict(np.array(X_test_seq))

pd.Series(X_pred_rnn[0]).describe()
```

Gambar 4. 14 Model LSTM (Long Short-Term Memory) lanjutan

Pesan dari TensorFlow yang memberi tahu bahwa alokasi memori yang diminta melebihi 10% dari memori sistem yang tersedia. Hal ini terjadi ketika model menggunakan ukuran batch yang besar atau data pelatihan memiliki dimensi yang besar. Dalam pesan tersebut, jumlah alokasi memori yang diminta adalah 12.000.000, yang setara dengan 12 juta unit memori. Pesan tersebut memperingatkan bahwa alokasi ini melebihi 10% dari total memori sistem yang tersedia.

Bukan kesalahan yang kritis, tetapi peringatan yang mengindikasikan bahwa penggunaan memori cukup tinggi. Memungkinkan berdampak pada kinerja yang kurang optimal atau bahkan kegagalan pelatihan jika memori tidak mencukupi.

Terdapat keterkaitan antara potongan kode yang ada dibawahnya. Potongan kode tersebut bertanggung jawab untuk memuat data uji (test data) dari file-file dalam direktori 'test' dan menyusunnya ke dalam format yang sesuai untuk digunakan dalam model kode kami.

(**X_train_seq** = []) digunakan untuk mempersiapkan data pelatihan (training data) dengan cara yang serupa dilakukan sebelumnya untuk data uji. Kemudian membaca data dari file 'train.csv' dalam potongan-potongan (chunks) yang sesuai, lalu mengonversinya ke dalam format yang sesuai untuk digunakan dalam model LSTM.

Potongan kode tersebut bertanggung jawab untuk melakukan proses serupa, tetapi pada data uji, membaca setiap file dalam direktori 'test', memuat data dari setiap file tersebut, dan kemudian mengonversinya ke dalam format yang sesuai dengan model sebelumnya. Dalam hal ini, list **X_test_seq** yang berisi setiap sampel data uji dalam format yang sesuai. Bisa kami simpulkan bahwa Metode LSTM tidak berhasil atau tidak cocok dalam kasus ini.

BAB V

KESIMPULAN

Berdasarkan hasil evaluasi pada data pengujian, kami menemukan bahwa model **'RandomForest'** dan **'SVR'** memberikan kinerja yang terbaik dalam memprediksi waktu tersisa hingga terjadinya gempa bumi laboratorium. Meskipun model **'LSTM'** dilatih menggunakan urutan 150000 tidak berhasil memberikan hasil yang memuaskan, dengan pelatihan yang memakan waktu dan Mean Absolute Error (MAE) yang besar. Namun, kami tetap akan berusaha menyempurnakan model **'RandomForest'** dan **'SVR'** lebih lanjut dengan melakukan penyesuaian parameter serta eksperimen tambahan. Dengan demikian, kami berharap dapat meningkatkan kinerja model dan membuat prediksi yang lebih akurat untuk memperbaiki penilaian risiko terhadap gempa bumi laboratorium di masa depan.

5.1 Proses Data Tes

```
x_test = pd.DataFrame()
seg_ids = pd.Series()

for file in os.listdir('test'):
    data = pd.read_csv(os.path.join('test', file), dtype = {'acoustic_data': np.int16})
    data_fet = gen_features(data['acoustic_data'])
    x_test = x_test.append(data_fet, ignore_index=True)
    seg_ids = seg_ids.append(pd.Series(os.path.splitext(file)[0]))

x_test_scaled = scaler.transform(x_test)
```

Gambar 5. 1 Kode Untuk Proses Data Tes

X_test = pd.DataFrame(): Membuat DataFrame kosong **X_test** yang akan digunakan untuk menyimpan fitur-fitur dari data uji.

seg_ids = pd.Series(): Membuat Series kosong **seg_ids** yang akan digunakan untuk menyimpan ID segmen dari setiap file data uji. Ini akan membantu dalam mengevaluasi prediksi yang dihasilkan oleh model.

data_fet = gen_features(data['acoustic_data']): Memanggil fungsi **gen_features()** untuk menghasilkan fitur-fitur statistik dari data akustik yang dibaca.

X_test = X_test.append(data_fet, ignore_index=True): Menambahkan fitur-fitur yang dihasilkan dari data akustik ke dalam **DataFrame X_test**. Setiap baris dalam DataFrame ini akan mewakili satu sampel data uji.

X_test_scaled = scaler.transform(X_test): Menggunakan scaler yang sudah didefinisikan sebelumnya untuk melakukan transformasi skala pada data uji. Skala transformasi dilakukan agar data uji memiliki skala yang sama dengan data pelatihan yang telah digunakan untuk melatih model.

5.2 Membuat Prediksi

```
forest_pred = rand_forest.predict(x_test_scaled)
svr_pred = svr.predict(x_test_scaled)

svr_pred

array([5.57793914, 4.90285036, 3.8717022 , ..., 4.97489095, 7.17243697,
       3.71733784])

forest_pred

array([4.60990848, 3.30905816, 3.31056604, ..., 3.96815423, 8.72599926,
       3.08890335])
```

Gambar 5. 2 Kode Untuk Prediksi Data

forest_pred = rand_forest.predict(X_test_scaled): Menggunakan model RandomForest yang telah dilatih sebelumnya (**rand_forest**) untuk membuat prediksi waktu tersisa hingga terjadinya gempa bumi pada data uji yang telah disiapkan (**X_test_scaled**). Prediksi tersebut disimpan dalam array **forest_pred**.

svr_pred = svr.predict(X_test_scaled): Menggunakan model SVR yang telah dilatih sebelumnya (**svr**) untuk membuat prediksi waktu tersisa hingga terjadinya gempa bumi pada data uji yang telah disiapkan (**X_test_scaled**). Prediksi tersebut disimpan dalam array **svr_pred**.

svr_pred: Menampilkan prediksi waktu tersisa hingga terjadinya gempa bumi yang dihasilkan oleh model SVR.

forest_pred: Menampilkan prediksi waktu tersisa hingga terjadinya gempa bumi yang dihasilkan oleh model RandomForest.

REFERENCES

- Heri Saputra, Muhammad Arsyad, dan Sulistiawaty. (2015). STUDI ANALISIS PARAMETER GEMPA DAN POLA SEBARAYA BERDASARKAN DATA MULTISTATION
<https://media.neliti.com/media/publications/319154-studi-analisis-parameter-gempa-dan-pola-8f9fc6a8.pdf>
- Yova Rezky Amanda, Mumtazah Nurul 'Aini, Melliyma Miyoz, dan Dwi Oktavianto Wahyu Nugroho. (2022). Prediksi Gempa Bumi di Indonesia Menggunakan R-Shiny.
https://ejurnal.its.ac.id/index.php/sains_seni/article/viewFile/62562/7080
- Zhen Wang. (2023). Seismic Signal Processing.
<https://csip.ece.gatech.edu/seismic-signal-processing/>
- Guntur Pasau dan Adey Tanauma (2011). PEMODELAN SUMBER GEMPA DI WILAYAH SULAWESI UTARA SEBAGAI UPAYA MITIGASI BENCANA GEMPA BUMI 1)
<https://media.neliti.com/media/publications/288446-pemodelan-sumber-gempa-di-wilayah-sulawesi-10ef0a8c.pdf>
- Gustavo Martins (2021). Predicting Earthquakes using Machine Learning
<https://medium.com/marionete/predicting-earthquakes-using-machine-learning-21689435dc52>
- Nabeel Ahmad, Shahid Riaz, Muhammad Awais. (2009). Earthquake Prediction: A global review and local research.
https://www.researchgate.net/publication/215837001_Earthquake_Prediction_A_global_review_and_local_research
- Robert J. Geller (1997). Earthquake prediction: a critical review
<https://academic.oup.com/gji/article/131/3/425/2138719>

LAMPIRAN

```
sub = pd.DataFrame(columns=['seg_id', 'time_to_failure'])
sub['seg_id'] = seg_ids
sub['time_to_failure'] = forest_pred
sub.head()
```

| | seg_id | time_to_failure |
|---|------------|-----------------|
| 0 | seg_5924f5 | 4.609908 |
| 0 | seg_5975f4 | 3.309058 |
| 0 | seg_455b16 | 3.310566 |
| 0 | seg_7e3b3e | 3.698824 |
| 0 | seg_e5fbc1 | 2.421940 |

Gambar 5. 3 Kode Untuk Hasil Prediksi Data

Output tersebut adalah hasil dari prediksi waktu tersisa hingga terjadinya gempa bumi yang dihasilkan oleh model ``RandomForest``.