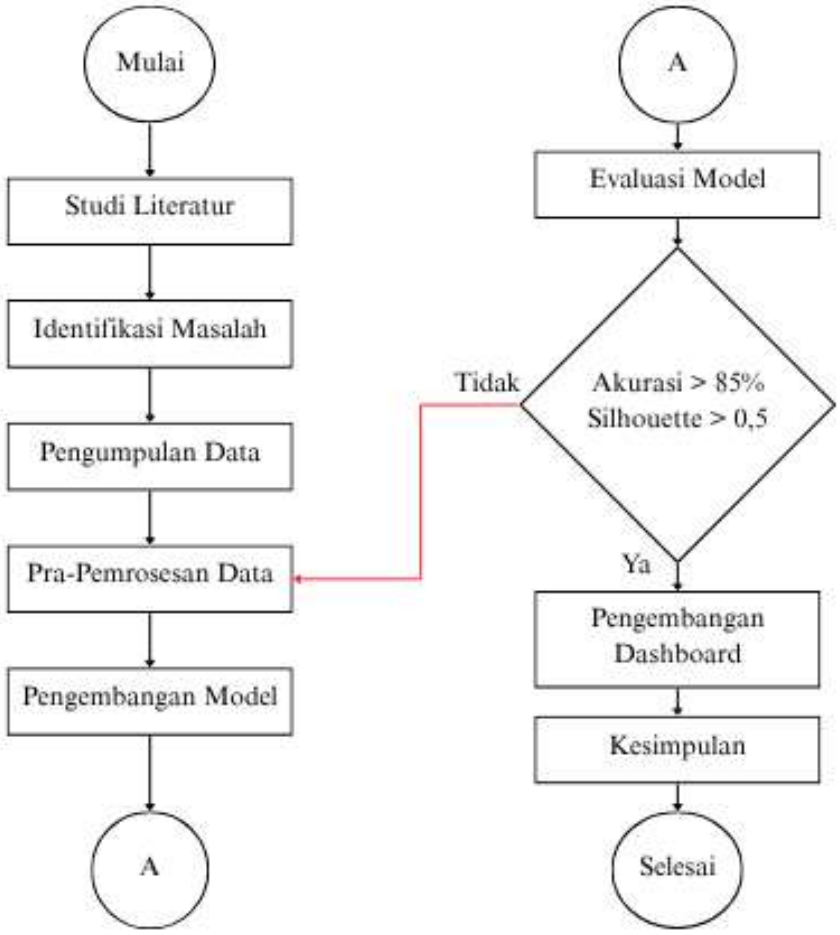
BAB III
METODOLOGI PENELITIAN

3.1 Metode dan Alur Penelitian

Metode penelitian yang dilakukan mengikuti pendekatan *research and development* (R&D) dengan alur penelitian yang ditunjukan pada Gambar 3.1.



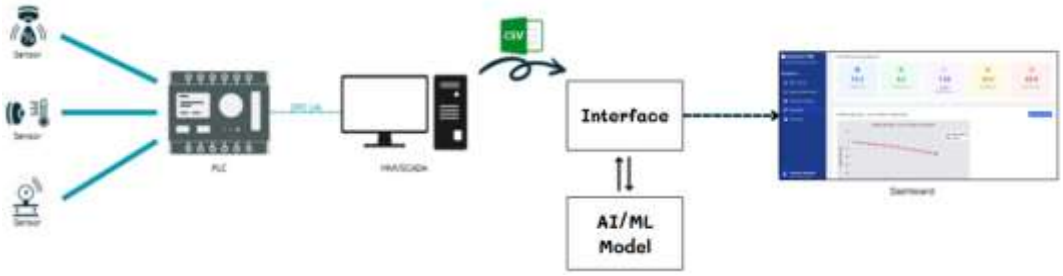
Gambar 3.1 *Flowchart* Alur Penelitian

3.2 Deskripsi Kerja Sistem

Mekanisme kerja sistem yang dirancang ditampilkan pada Gambar 3.2. Sistem ini memungkinkan pengguna untuk mengunggah data operasional dalam format CSV melalui antarmuka berbasis *website*, data yang telah diunggah akan diproses menggunakan model XGBoost untuk melakukan peramalan terhadap parameter operasional mesin, seperti konduktivitas, ketinggian air, dan tekanan. Model XGBoost akan menganalisis tren historis dan memprediksi nilai parameter

tersebut untuk periode waktu ke depan. Selanjutnya, data historis dan hasil peramalan dari model XGBoost dianalisis menggunakan *isolation forest* untuk mendeteksi anomali. Anomali yang terdeteksi dapat menunjukkan potensi masalah atau tanda-tanda awal kegagalan mesin sebelum terjadi kerusakan serius. Setelah itu, data historis kegagalan digunakan untuk melatih model *random forest*, model ini bertugas untuk memprediksi kemungkinan kegagalan mesin pada data operasional dan data ramalan dengan cara memberikan label apakah suatu kondisi memiliki risiko tinggi terhadap kegagalan atau tidak.

Hasil dari seluruh proses ini kemudian divisualisasikan dalam dashboard dengan bentuk grafik interaktif dan tabel, sehingga pengguna dapat dengan mudah memahami tren data, titik anomali, serta prediksi kegagalan mesin. Selain itu, sistem ini juga memungkinkan pengguna untuk mengunduh hasil analisis dalam format CSV untuk keperluan lebih lanjut.



Gambar 3.2 Alur Kerja Sistem

3.3 *Data Acquisition*

Dalam penelitian ini, data diperoleh dari sensor konduktivitas, level air, dan tekanan pada *electric boiler* di PT Vale Indonesia selama rentang waktu 4 Februari 2024 hingga 18 Februari 2024. Data dikumpulkan dalam bentuk *time series* dengan interval pencatatan setiap detik selama dua minggu, sehingga menghasilkan dataset yang cukup untuk analisis tren jangka panjang, deteksi anomali, dan prediksi kegagalan. Sensor konduktivitas digunakan untuk mengukur tingkat konduktivitas air dalam *electric boiler* guna mendeteksi perubahan kualitas air yang dapat mempengaruhi efisiensi dan umur mesin. Sensor level air berfungsi untuk memantau ketinggian air dalam *electric boiler* agar tetap dalam kondisi optimal serta menghindari risiko kegagalan akibat level air yang terlalu rendah atau tinggi. Sementara itu, sensor tekanan digunakan untuk mengukur tekanan uap dalam

electric boiler guna mendeteksi fluktuasi abnormal yang berpotensi menjadi indikator awal kegagalan sistem.

Struktur data yang dikumpulkan terdiri dari beberapa fitur utama, yaitu *timestamp* yang mencatat waktu pengambilan data dalam format YYYY-MM-DD HH:MM:SS, nilai konduktivitas air dalam satuan mikrosiemens per sentimeter (µS/cm), tinggi air dalam satuan sentimeter (cm), dan tekanan dalam satuan bar. Setelah dikumpulkan, data ini akan melalui proses *pre-processing* yang mencakup, penanganan data duplikat, penanganan *missing values* menggunakan metode mean, *resampling* untuk menurunkan tingkat *noise*, dan metode *principle components analysis* untuk menghilangkan bias antara dua label data yang berbeda.

Selain data sensor, data kegagalan juga dikumpulkan sebagai referensi dalam proses pelabelan dan validasi sistem prediktif. Peristiwa kegagalan ini diperoleh dari catatan sistem pemeliharaan *electric boiler* di PT Vale Indonesia, yang terdokumentasi secara sistematis dalam laporan kerja dan histori kejadian. Salah satu kejadian kegagalan yang digunakan dalam penelitian ini adalah insiden dengan deskripsi "E. BOILER CWP#B SEALING LINE LEAKS" yang terjadi pada tanggal 29 April 2024. Peristiwa ini tercatat terjadi pada lokasi fungsi "THERMAL EL BOILER CIRCULATING PUMP #2". Data kejadian tersebut digunakan sebagai titik acuan dalam mengidentifikasi pola-pola parameter sensor selama kegagalan.

3.4 *Data Pre-Processing*

3.4.1 *Data Labelling*

Dalam proses pelabelan data, tujuan utama adalah membedakan antara data normal dan data yang menunjukkan indikasi kegagalan pada *electric boiler*. Pelabelan ini dilakukan dengan mengacu pada catatan peristiwa kegagalan yang terdokumentasi dalam sistem pemeliharaan PT Vale Indonesia. Salah satu kejadian kegagalan yang dijadikan referensi dalam penelitian ini adalah peristiwa dengan deskripsi "E. BOILER CWP#B SEALING LINE LEAKS" yang terjadi pada tanggal 29 April 2024 dengan lokasi fungsi "THERMAL EL BOILER CIRCULATING PUMP #2".

Untuk melakukan pelabelan, data historis dari sensor konduktivitas, level air, dan tekanan dianalisis berdasarkan informasi waktu kegagalan yang telah tercatat. Data yang berada dalam rentang waktu yang sesuai dengan informasi kegagalan dikategorikan sebagai data kegagalan, sedangkan data di luar periode tersebut yang tidak menunjukkan pola abnormal dikategorikan sebagai data normal. Proses pelabelan dilakukan dengan metode berikut:

1. Identifikasi waktu kegagalan: data sensor dianalisis berdasarkan waktu kejadian kegagalan, seperti yang tercatat dalam laporan operasional.
2. Pemberian label kegagalan: data yang berada dalam rentang waktu yang sesuai dengan informasi kegagalan diberikan label nilai 1.
3. Pemberian label normal: data yang berada di luar periode kegagalan dan tidak menunjukkan pola perubahan signifikan diberi label nilai 0.

3.4.2 *Data Cleaning*

Pada tahap ini, dilakukan proses pembersihan data untuk memastikan kualitas dan konsistensi data sebelum digunakan dalam pelatihan model *machine learning*. Proses dimulai dengan menghapus data duplikat, yaitu data yang memiliki nilai identik pada kolom waktu dan tanggal, konduktivitas, tekanan, dan tinggi air, karena dianggap tidak memberikan informasi tambahan dan dapat mengganggu proses pembelajaran model. Selanjutnya, dilakukan penanganan terhadap nilai kosong atau *missing values* pada fitur numerik. Nilai-nilai kosong ini diimputasi menggunakan metode rata-rata atau *mean* dari masing-masing kolom sensor agar distribusi data tetap konsisten dan tidak menyebabkan bias.

Dalam konteks deteksi anomali menggunakan model *isolation forest*, proses pra-pemrosesan data tidak menghilangkan *outlier*, karena nilai-nilai yang menyimpang ini justru menjadi indikator penting dalam mendeteksi perilaku abnormal atau gangguan sistem, termasuk yang disebabkan oleh kejadian eksternal seperti *blackout*. Oleh karena itu, data dengan *outlier* tetap dipertahankan agar model dapat mempelajari pola-pola anomali yang relevan.

Sebaliknya, untuk pelatihan model XGBoost yang ditujukan untuk memprediksi data sistem berdasarkan pola operasi normal, dilakukan penghapusan *outlier*. *Outlier* yang terdeteksi sebagian besar disebabkan oleh *blackout*, yaitu

gangguan dari luar sistem yang tidak merefleksikan kinerja internal *electric boiler*. Oleh karena itu, untuk menjaga akurasi dan relevansi prediksi, data *outlier* ini dihapus dari dataset pelatihan XGBoost.

3.4.3 *Data Resampling*

Data dari sistem *electric boiler* direkam setiap detik, menghasilkan lebih dari satu juta entri per sensor. Meskipun resolusi tinggi ini memberikan informasi yang sangat detail, volume data yang besar dapat menyebabkan beban komputasi tinggi dan meningkatkan *noise*. Untuk mengatasi hal ini, dilakukan proses *resampling* dengan agregasi rata-rata dengan interval *resampling* diperluas menjadi per jam. Pendekatan ini bertujuan menyeimbangkan jumlah data dan kualitas prediksi, sekaligus fokus pada tren jangka pendek hingga menengah yang lebih relevan dengan perubahan operasional secara makro. Dengan *resampling* per jam, data menjadi lebih halus dan stabil, sehingga model lebih mudah mengenali pola penting tanpa terganggu fluktuasi kecil yang tidak signifikan. Selain itu, hasil prediksi per jam juga lebih ringkas dan efisien yang mana cukup satu prediksi untuk satu jam operasional, sehingga memudahkan analisis dan interpretasi.

3.5 Pemodelan *Machine Learning* untuk Pemeliharaan Prediktif

3.5.1 Pemisahan Data *Training* dan *Testing*

Dalam proses ini, dataset dibagi untuk membangun model pembelajaran mesin. Data *training* digunakan untuk melatih model agar memperoleh akurasi yang optimal dalam melakukan klasifikasi dan ramalan. Sementara itu, data testing digunakan untuk mengevaluasi akurasi model pada dataset yang tersedia. Pada penelitian ini, pembagian *data training* dan *testing* dilakukan dengan rasio 80:20, yang mana *data training* memiliki jumlah data yang lebih besar.

3.5.2 Membangun Model *Extreme Gradient Boosting* (XGBoost)

XGBoost merupakan algoritma *supervised learning* berbasis *gradient boosting* yang digunakan untuk tugas regresi. Metode ini bekerja dengan membangun pohon keputusan secara bertahap dan mengoptimalkan kesalahan prediksi menggunakan teknik gradien. Berikut adalah langkah-langkah dalam membangun model XGBoost untuk peramalan:

1. Tentukan variabel dan parameter pada model

   $x = \{x_1, x_2, \ldots, x_l\}$: sampel *training*

$y = \{y_1, y_2, \ldots, y_l\}$: label data *training*

2. Membangun model peramalan menggunakan *library* XGBoost, yaitu xgboost.XGBRegressor dengan bahasa pemrograman Python pada *software* Google Colab
3. Lakukan *hyperparameter tuning* menggunakan GridSearchCV dengan parameter:
   - n_estimators: jumlah pohon keputusan yang akan dibangun
   - max_depth: kedalaman maksimum pohon untuk menghindari *overfitting*
4. Evaluasi performa model *forecasting* dengan memperhitungkan *error* dengan metode *mean absolute percentage error* (MAPE) melalui *library* sklearn.metrics dari Python, yang menyediakan fungsi mean_absolute_percentage_error()

3.5.3 Membangun Model *Isolation Forest* (IForest)

Isolation forest adalah model *unsupervised learning* yang digunakan untuk mendeteksi anomali dalam suatu dataset. Algoritma ini bekerja dengan membangun sekumpulan *isolation trees* (iTrees) untuk memisahkan data berdasarkan pemotongan acak. Data yang lebih cepat terisolasi cenderung dianggap sebagai anomali, sedangkan data yang sulit dipisahkan dianggap sebagai data normal. Berikut adalah langkah-langkah dalam membangun model *isolation forest* untuk deteksi anomali:

1. Tentukan variable dan parameter pada model
   $x = \{x_1, x_2, \ldots, x_l\}$: sampel *training*
2. model deteksi anomali menggunakan *library* Scikit-Learn (sklearn) dengan sklearn.ensemble.IsolationForest pada *software* Google Colab dengan bahasa pemrograman Python
3. Lakukan *hyperparameter tuning* menggunakan GridSearchCV dengan parameter:
   - n_estimators: jumlah pohon dalam model *isolation forest*. Semakin banyak pohon, semakin baik deteksi anomali, tetapi juga lebih lambat.

- contamination: Proporsi data yang dianggap anomali. Jika "auto", model akan memperkirakan sendiri. Jika diisi angka (misalnya 0.1), berarti 10% data dianggap anomali.

4. Evaluasi performa model deteksi anomali dilakukan menggunakan nilai *silhouette score* yang dihitung melalui *library* sklearn**.**metrics pada Python.

3.5.4 Membangun Model *Random Forest* (RF)

Sedangkan *random forest* adalah model *supervised learning* yang menggunakan kumpulan dari *decision tree* untuk melakukan prediksi klasifikasi. Berikut adalah langkah-langkah dalam membangun model *random forest* untuk prediksi kegagalan:

1. Tentukan variabel dan parameter pada model
   $x = \{x_1, x_2, \ldots, x_l\}$: sampel *training*
   $y = \{y_1, y_2, \ldots, y_l\}$: label data *training*
2. Buat model prediksi dengan menggunakan *library* sklearn yaitu RandomForestClassifier pada *software* Google Colab dengan bahasa pemrograman Python.
3. Lakukan *hyperparameter tuning* menggunakan GridSearchCV dengan parameter:

   - n_estimators: Kedalaman maksimal setiap pohon. Jika terlalu dalam, model bisa terlalu rumit dan *overfitting*, jika terlalu dangkal, model bisa kurang akurat atau *underfitting*
   - max_depth: Kedalaman maksimum setiap pohon. Jika terlalu dalam, model bisa terlalu kompleks dan mengalami *overfitting*. Sebaliknya, jika terlalu dangkal, model bisa gagal menangkap kompleksitas data *underfitting*
   - min_samples_split: Jumlah minimum data yang dibutuhkan agar suatu cabang dalam pohon bisa dibagi lagi. Jika terlalu kecil, pohon bisa jadi terlalu kompleks
   - criterion: Fungsi untuk mengukur kualitas split. Untuk model ini, opsi umum adalah "gini" atau "entropy". Pemilihan tergantung pada

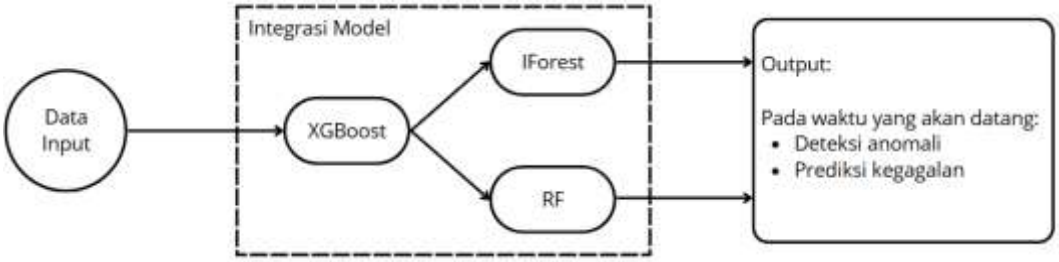
preferensi dan performa empiris, namun keduanya umumnya memberikan hasil yang mirip.

4. Evaluasi peforma model klasifikasi dengan *confusion matrix*, memperhitungkan *accuracy*, *precision*, *recall*, f1-*score*, dan AUC dari ROC *curve* melalui *library* sklearn.metrics dengan bahasa pemrograman Python

3.6 Integrasi Model

Integrasi model bekerja dapat dilihat pada Gambar 3.3. Dalam integrasi model, data yang diperoleh dari sensor dan sudah melalui tahap *data preprocessing* akan dianalisis menggunakan model XGBoost untuk melakukan peramalan data berdasarkan pola data historis.

Selanjutnya, hasil ramalan dari model XGBoost akan diolah dengan model *isolation forest* dan *random forest*, pada proses ini menggunakan dua data, yakni data aktual dan hasil peramalan. *Isolation forest* berfungsi untuk deteksi anomali, yaitu mengidentifikasi data yang menyimpang dari pola normal. Sementara itu, model *random forest* digunakan untuk prediksi potensi kegagalan berdasarkan data historis kegagalan yang pernah terjadi sebelumnya.



Gambar 3.3 Integrasi Model

3.7 Perancangan Dashboard

Perancangan dashboard dilakukan dengan memanfaatkan teknologi HTML, CSS, dan JavaScript untuk pengembangan *frontend*, serta *framework* Flask pada bagian *backend* yang bertugas menjalankan model machine learning. Perancangan dilakukan secara bertahap untuk memastikan integrasi antara tampilan antarmuka dan model prediksi berjalan optimal.

a. Perancangan *frontend*

Antarmuka pengguna dirancang agar responsif, intuitif, dan memudahkan teknisi dalam memantau kondisi mesin. Beberapa elemen penting pada perancangan *frontend* antara lain:

1. Sidebar navigasi menyediakan navigasi ke tiga *tab* utama, yaitu:
    - *Data upload*
    - *Real-time monitoring*
    - *Predictive analytics*
2. *Tab data upload* disediakan file *upload box* yang memungkinkan pengguna mengunggah data operasional mesin. *Box* ini dilengkapi keterangan mengenai format file dan persyaratan data yang harus dipenuhi agar sistem dapat memprosesnya dengan benar.
3. *Tab real-time monitoring* menampilkan grafik visualisasi data sensor secara *real-time* untuk membantu pengguna memantau parameter mesin. Grafik ini diintegrasikan dengan model *isolation forest* pada *backend* untuk mendeteksi anomali secara *real-time*. Titik data yang terdeteksi sebagai anomali akan ditandai dengan warna merah untuk memudahkan identifikasi visual.
4. *Tab predictive analytics* dirancang lebih komprehensif dengan fitur:
    - Grafik *forecasting* menggunakan model XGBoost untuk memprediksi tren data operasional.
    - Deteksi anomali dengan penandaan plot merah pada data yang terdeteksi oleh model *isolation forest*.
    - Tabel prediksi kegagalan yang menampilkan hasil prediksi model *random forest* terkait potensi kegagalan mesin.
    - *Bar chart* dan *pie chart* untuk menampilkan performa mesin berdasarkan proporsi data normal, data anomali, dan data kegagalan yang terdeteksi.
    - Tabel *alert history* yang mencatat seluruh notifikasi anomali maupun kegagalan yang pernah terdeteksi, termasuk waktu kejadian dan deskripsi masalah.

b. Perancangan *backend*

Perancangan backend dilakukan dengan memanfaatkan framework Flask untuk menangani komunikasi antara *frontend*, database, dan model machine learning. Backend berfungsi untuk:

1. Mengintegrasikan model-model machine learning yang digunakan, antara lain:
    - XGBoost untuk forecasting data operasional.
    - Isolation Forest untuk deteksi anomali secara real-time.
    - Random Forest untuk prediksi kegagalan mesin.
2. Mengelola data sensor yang diterima secara real-time dan memastikan hasil prediksi dapat diakses dengan cepat oleh *frontend*.
3. Menyimpan data alert ke dalam basis data agar dapat ditampilkan pada tabel *Alert History*.