PREDIKSI SUHU MAGNET PERMANEN PADA MOTOR LISTRIK MENGGUNAKAN ALGORITMA *SUPERVISED REGRESSION*

LAPORAN

disusun untuk memenuhi tugas mata kuliah TF4063

Sains Data Rekayasa pada semester II tahun ajaran 2021/2022

oleh

Akhmad Aji Permadi 10118030

Arya Pratama Putra 10118104

PROGRAM STUDI TEKNIK FISIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Analisis regresi adalah konsep mendasar di bidang machine learning. Regresi termasuk dalam supervised learning di mana model dilatih dengan fitur input dan label output. Regresi membantu dalam membangun hubungan antara variabel dengan memperkirakan bagaimana satu variabel mempengaruhi yang lain. Dengan regresi, kita dapat membuat prediksi suatu variabel respons/target menggunakan data-data dari variabel prediktor.

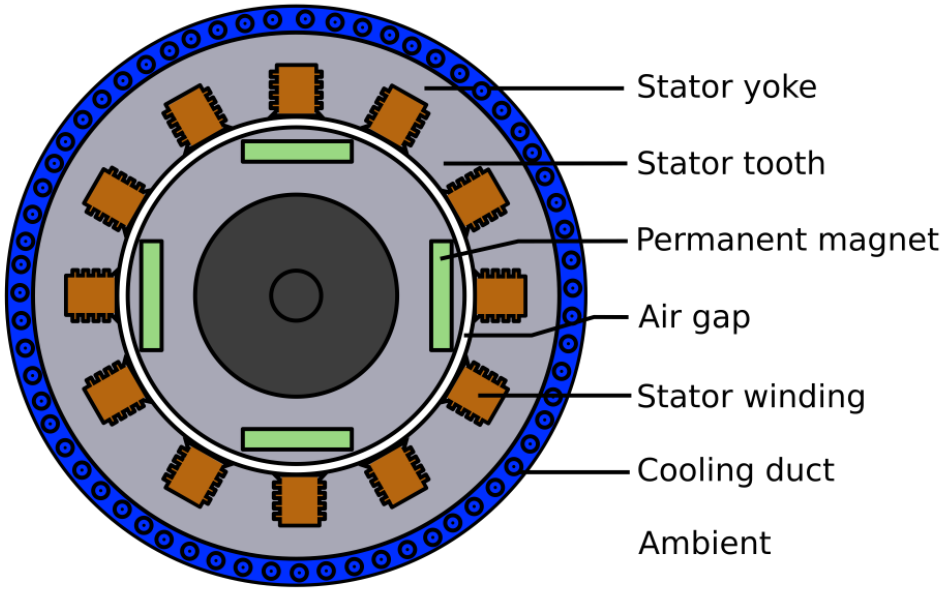
Di sebagian besar aplikasi otomotif, daya tinggi dan densitas torsi serta efisiensi diperlukan untuk setiap motor tertentu, yang membuat Permanent-Magnet Synchronous Machine (PMSM) menjadi pilihan yang lebih disukai. Untuk memanfaatkan kemampuan mesin sepenuhnya, tekanan termal yang tinggi pada komponen mesin yang berpotensi gagal harus diperhitungkan.

Pengukuran suhu berbasis sensor akan menghasilkan pengetahuan yang tepat mengenai kondisi termal mesin. Namun untuk bagian rotor, secara teknis dan ekonomis tidak layak karena struktur internal motor listrik yang canggih dan aksesibilitas yang sulit dari rotor. Oleh karena itu, teknik monitoring rotor langsung seperti infrared thermography atau classic thermocouples dengan cincin selip yang dipasang di poros gagal memasuki seri produksi industri. Sebaliknya, stator winding temperature monitoring di masa kini diukur pada sensor dasar, namun dalam kasus kesalahan, sensor ini tidak dapat diganti karena tertanam kuat di stator. Selain itu, fungsi sensor dapat menurun selama masa pengoperasian motor. Mengingat semakin pentingnya yang keselamatan fungsional terutama di industri otomotif, informasi suhu yang berlebihan menjadi wajib.

Pendekatan tipikal untuk memperkirakan suhu komponen internal PMSM terdiri dari pengaturan model mesin listrik yang memberikan informasi tentang parameter model listrik yang sensitif terhadap suhu secara tidak langsung. Ada metode yang bekerja dengan injeksi arus atau injeksi tegangan untuk mendapatkan stator winding resistanc atau tingkat magnetisasi dari magnet sebagai indikator termal. Selain itu, fundamental wave flux observer dapat membantu menilai demagnetisasi reversibel dari magnet

tertanam. Namun, metode ini adalah model dengan sensitivitas parameter model listrik yang tinggi, sehingga usaha pemodelan yang tidak mencukupi menyebabkan kesalahan estimasi yang berlebihan.

Dalam tugas ini, akan ditunjukkan bahwa model machine learning regresi menghindari segala jenis pemodelan motor listrik dengan dipasang pada data test bench terukur dengan preprocessing kecil secara langsung dan masih mencapai kinerja estimasi yang baik dengan properti time invariant.



Gambar 1. Ilustrasi Electric Motor

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Model apa saja yang dapat digunakan untuk memprediksi suhu magnet permanen?
2. Bagaimanakah perbandingan performa model dalam memprediksi suhu magnet permanen?
3. Bagaimanakah perbandingan efisiensi model dalam memprediksi Suhu magnet permanen?
4. Manakah model yang terbaik untuk memprediksi Suhu magnet permanen?

1.3 Tujuan Penelitian

Tujuan yang hendak dicapai melalui penelitian ini adalah sebagai berikut.

1. Mengetahui model-model *machine learning* yang dapat digunakan untuk memprediksi suhu magnet permanen pada motor listrik.
2. Membandingkan model prediksi suhu magnet permanen pada motor listrik berdasarkan performa.
3. Membandingkan model prediksi suhu magnet permanen pada motor listrik berdasarkan efisiensi.
4. Mengetahui model terbaik dalam memprediksi suhu magnet permanen pada motor listrik.

1.4 Manfaat Penelitian

Penelitian ini dibuat dengan harapan dapat memberikan manfaat bagi pihak manapun yang membutuhkan, baik manfaat secara teoritis maupun praktis, di antaranya:

1. Manfaat teoritis

   Penelitian ini diharapkan dapat menambah wawasan dan pengetahuan penulis maupun pembaca mengenai beberapa jenis metode yang dapat digunakan dalam melakukan prediksi suhu magnet permanen pada motor listrik, sekaligus mengetahui performa serta efisiensi dari setiap metode yang digunakan.
2. Manfaat praktis
   a. Bagi penulis Penelitian ini diharapkan dapat menjadi sarana untuk mengimplementasikan wawasan dan pengetahuan penulis mengenai *machine learning* yang diperoleh dari kelas Sains Data Rekayasa sekaligus dari sumber-sumber yang lain.
   b. Bagi peneliti selanjutnya Penelitian ini diharapkan dapat dijadikan sebagai bahan referensi sekaligus memberikan kontribusi dalam pengembangan model-model untuk mendapatkan tingkat performa dan efisiensi yang lebih baik.

BAB II
METODOLOGI

2.1 Metode Acuan

Program yang penulis gunakan sebagai acuan adalah program yang dibuat oleh Pratik Asarkar yang dapat dilihat pada tautan berikut. Program acuan tersebut menggunakan dataset dari kaggle. Kaggle merupakan situs yang menyediakan berbagai macam dataset yang dapat digunakan oleh publik. Berikut adalah penjelasan dari metode acuan yang penulis gunakan.

2.1.1 Dataset

Dataset yang digunakan terdiri dari beberapa data sensor yang dikumpulkan dari PMSM yang ditempatkan pada *test bench*. PMSM yang digunakan merepresentasikan model prototipe OEM Jerman. Pengukuran *test bench* dikumpulkan oleh departemen LEA di Universitas Paderborn. Dataset ini sedikit disamarkan.

Semua perekaman diambil sampelnya pada 2 Hz. Dataset terdiri dari beberapa sesi pengukuran, yang dapat dibedakan satu sama lain dengan kolom "profile_id". Sesi pengukuran dapat berlangsung antara satu hingga enam jam. Motor dieksitasi dengan hand-designed driving cycles yang menunjukkan referensi kecepatan motor dan referensi torsi. Arus dalam koordinat d/q (kolom "id" dan iq") dan tegangan pada koordinat d/q (kolom "ud" dan "uq") adalah hasil dari strategi kontrol standar yang mencoba mengikuti referensi kecepatan dan torsi. Kolom "motor_speed" dan "torque" adalah kuantitas hasil yang dicapai oleh strategi itu, yang diturunkan dari arus dan tegangan yang ditetapkan.

Sebagian besar driving cycles menunjukkan random walks di bidang kecepatan-torsi untuk meniru driving cycles dunia nyata ke tingkat yang lebih akurat daripada eksitasi konstan serta peningkatan dan penurunan konstan. Dataset yang digunakan bukan merupakan versi terbaru, melainkan dataset *version 2* dengan nama file pmsm_temperature_data.csv dan diunggah pada 3 tahun lalu. Berikut ditunjukkan lima baris pertama pada dataset ini.

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding	profile_id
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.018033	4
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2.017631	4
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2.017343	4
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2.017632	4
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2.018145	4

Gambar 2. Sampel data yang digunakan

Dataset ini terdiri dari 998.070 baris dan 13 kolom dengan setiap kolomnya berisi nilai bilangan integer/float. Adapun ketigabelas kolom tersebut meliputi:

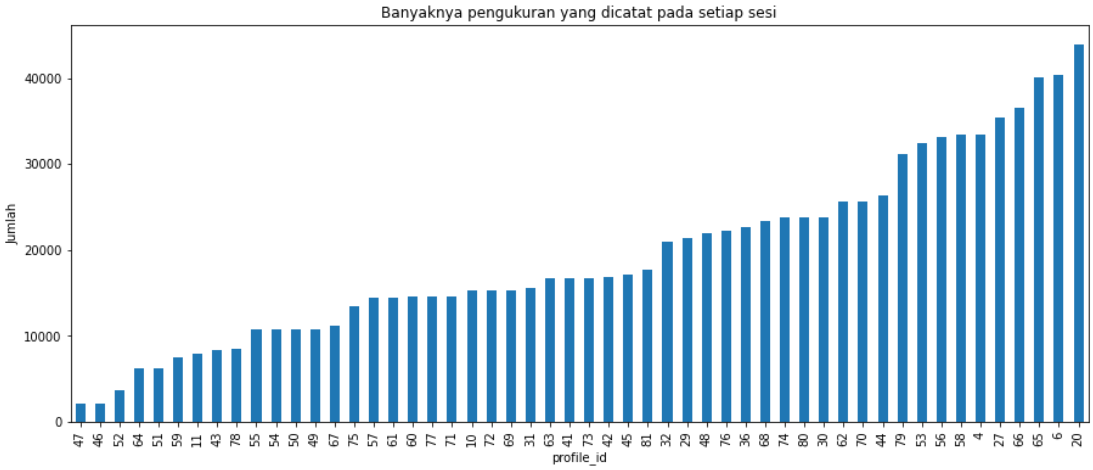
Kolom	**Deskripsi**
u_q	Pengukuran tegangan komponen q dalam koordinat dq (V)
coolant	Suhu pendingin ($°C$)
stator_winding	Suhu stator winding ($°C$) diukur dengan termokopel
u_d	Pengukuran tegangan komponen d dalam koordinat dq (V)
stator_tooth	Suhu stator tooth ($°C$) diukur dengan termokopel
motor_speed	Kecepatan motor (rpm)
i_d	Pengukuran arus komponen d dalam koordinat dq (A)
i_q	Pengukuran arus komponen q dalam koordinat dq (A)
pm	Suhu magnet permanen ($°C$) diukur dengan termokopel dan ditransmisikan secara nirkabel melalui unit termografi
stator_yoke	Suhu stator yoke ($°C$) diukur dengan termokopel
ambient	Suhu udara di sekitar motor ($°C$)
Torque	Torsi motor (Nm)
profile_id	ID sesi pengukuran. Setiap sesi pengukuran yang berbeda dapat diidentifikasi melalui ID ini

Tabel 1. Deskripsi kolom pada data

2.1.2 Explotarory Data Analysis

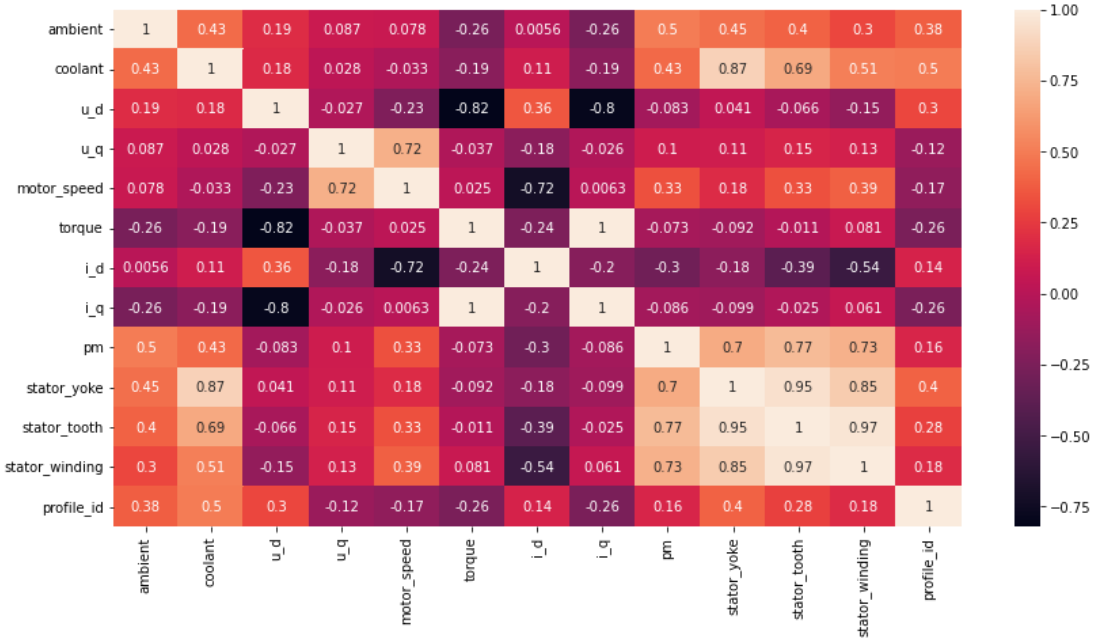
Pada bagian ini, dataset yang dimiliki akan dilakukan proses analisis data eksplorasi. EDA dilakukan dengan cara melakukan visualisasi terhadap dataset menggunakan library matplotlib pada python. Berikut ditunjukkan beberapa grafik yang akan memberikan informasi dari data yang dimiliki.

Grafik pertama adalah grafik yang menunjukkan banyaknya pengukuran yang dicatat pada setiap sesinya. Terlihat pada grafik di bawah, sesi dengan profil id 20 memiliki catatan pengukuran paling banyak, yang kemudian diikuti oleh sesi dengan profil id 6 dan 65.



Gambar 3. Banyaknya pengukuran yang dicatat pada setiap profile id

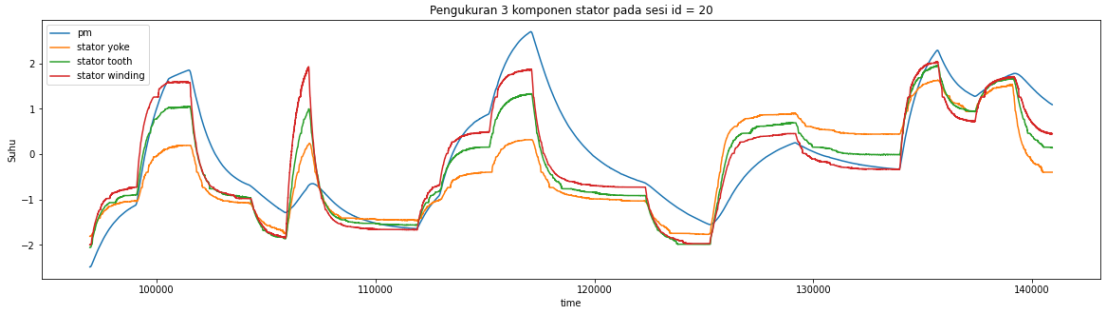
Selanjutnya, ditunjukkan sebuah heatmap yang memperlihatkan tingkat korelasi antara satu fitur dengan fitur lainnya. Dari heatmap di bawah ini terlihat bahwa torsi motor dengan fitur i_q memiliki korelasi yang terbilang sempurnya. Selain itu, terlihat juga bahwa adanya korelasi yang cukup tinggi di antara fitur stator_winding, stator_tooth, dan stator_yoke.



Gambar 4. Korelasi antar variabel

Grafik berikutnya adalah grafik yang menunjukkan perbandingan suhu dari ketiga komponen stator dan magnet permanen di setiap pengukuran. Pada kasus ini, diambil pengukuran pada sesi dengan profile_id = 20, yang merupakan sesi dengan catatan pengukuran terbanyak. Terlihat bahwa kecenderungan dari ketiga komponen

stator dan magnet permanen memiliki pola yang hampir sama antara satu dengan yang lainnya.



Gambar 5. Pengukuran 3 komponen stator dan magnet permanen pada sesi id = 20

2.1.3 Data Preprocessing

Pada bagian ini akan dilakukan beberapa preprocessing pada dataset yang dimiliki sebelum dataset siap digunakan pada pemodelan. Adapun ada beberapa hal yang dilakukan pada bagian ini, yaitu:

- Ambil beberapa data untuk data test

  Di sini akan diambil semua data pengukuran pada sesi dengan profile_id = 65 dan profile_id = 72.

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque	profile_id
984807	-1.887356	30.721162	32.113178	1.946434	31.291477	0.022686	-2.001102	1.098022	37.112483	30.660012	23.886441	0.000187	72
984808	0.604919	30.721209	32.115623	0.109622	31.296847	27.444022	-8.944310	26.184500	37.111457	30.708855	23.885538	18.702919	72
984809	4.665253	30.721242	32.122736	-6.753488	31.283346	116.920677	-35.659968	80.725465	37.116662	30.743853	23.883657	63.775405	72
984810	9.206368	30.721266	32.130268	-16.699363	31.283340	251.384409	-59.658771	125.454513	36.924837	30.777050	23.880380	101.423441	72
984811	14.197583	30.720804	32.117775	-28.559625	31.212032	418.588814	-77.320911	158.080952	36.922961	30.820728	23.874869	128.924817	72

Gambar 6. Sampel data setelah drop profile_id = 65 dan profile_id = 72

- Drop kolom profile_id

  Penghapusan kolom profil_id di dataset dilakukan karena fitur tersebut dianggap tidak terlalu berpengaruh.

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197

Gambar 7. Sampel data setelah kolom profile_id dihapus

- Drop beberapa kolom lainnya dan transformasi fitur

  Fitur suhu magnet permanen (p_m) akan dijadikan sebagai fitur target, sedangkan fitur lainnya akan dijadikan sebagai fitur predictor. Namun, pada bagian ini tidak semua fitur selain p_m akan menjadi fitur predictor, melainkan ada beberapa fitur yang tidak akan digunakan (yang berarti akan di-drop). Fitur-fitur lain yang tidak akan digunakan sebagai fitur predictor maupun target meliputi fitur: stator_yoke, stator_tooth, stator_winding, dan torsi. Pertimbangan menghilangkan ketiga komponen stator dikarenakan ketiga komponen stator tersebut memiliki korelasi yang terbilang tinggi dengan fitur target (pm), dapat dilihat dari grafik pada gambar 3 dan heatmap pada gambar 4. Sedangkan untuk torsi dilakukan drop dikarenakan dianggap tidak dapat diukur secara baik di lapangan. Selanjutnya fitur-fitur yang digunakan sebagai predictor akan transformasi scaling menggunakan MinMaxScaler.

2.1.4 Metode Multiple Linear Regression

Dalam statistika, regresi linier adalah pendekatan linier untuk memodelkan hubungan antara respons skalar (variabel dependen) dan satu atau lebih variabel *explanatory*. Kasus paling sederhana dari variabel prediktor skalar tunggal x dan variabel respons skalar tunggal y dikenal sebagai regresi linier sederhana. Perluasan ke variabel prediktor bernilai ganda dan/atau vektor (dilambangkan dengan huruf kapital X) dikenal sebagai regresi linier berganda.

Regresi linier berganda adalah generalisasi dari regresi linier sederhana untuk kasus lebih dari satu variabel independen, dan kasus khusus model linier umum, terbatas pada satu variabel dependen. Diberikan kumpulan data $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^{n}$ dari n unit statistik, model regresi linier mengasumsikan bahwa hubungan antara variabel dependen y dan vektor-p dari regressor x adalah linier. Hubungan ini dimodelkan melalui variabel error (ε), yaitu variabel acak yang tidak teramati yang menambahkan "gangguan" pada hubungan linier antara variabel dependen dan regresi. Dengan demikian model mengambil bentuk

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots \beta_p x_{ip} + \varepsilon_i$$

untuk setiap observasi $i = 1,2, \dots, n$. Persamaan ini juga dapat dinyatakan dalam bentuk berikut.

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$$

di mana

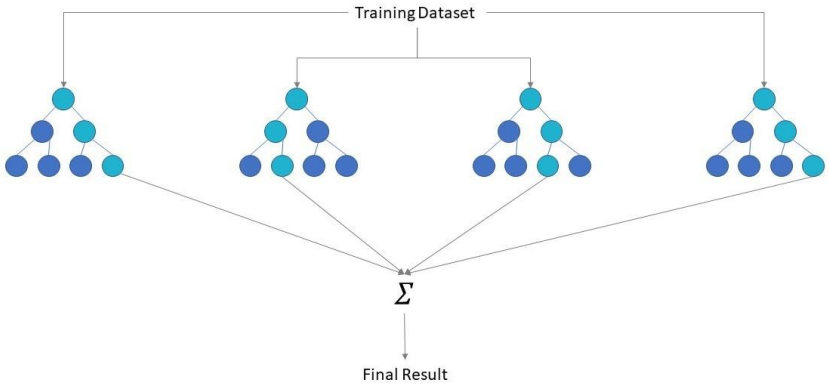
$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Dalam rumus di atas kita mempertimbangkan *n* pengamatan dari satu variabel dependen dan p variabel bebas. Jadi, y_i adalah pengamatan ke-i terhadap variabel dependen, x_{ij} adalah pengamatan ke-i terhadap variabel bebas ke-j, dengan $j = 1, 2, \ldots, p$. Nilai j mewakili parameter yang akan diestimasi, dan i adalah galat normal independen berdistribusi identik ke-i.

2.1.5 Metode Random Forest Regression

Random Forest adalah metode ensemble learning yang dapat digunakan untuk tugas klasifikasi atau regresi. Random forest beroperasi dengan membangun banyak decision tree pada saat training. Decision tree adalah classifier berstruktur pohon dengan tiga jenis node: root node, interior node dan leaf node. Root node adalah node awal yang mewakili seluruh sampel dan dapat dibagi lebih lanjut menjadi node selanjutnya. Interior node mewakili fitur dari kumpulan data dan cabang mewakili aturan keputusan. Sedangkan leaf node mewakili hasil. Dengan titik data tertentu, decision tree dijalankan sepenuhnya melalui seluruh pohon dengan menjawab pertanyaan benar/salah hingga mencapai simpul daun. Prediksi terakhir adalah rata-rata dari nilai variabel dependen dalam simpul daun tertentu. Melalui beberapa iterasi, decision tree mampu memprediksi nilai yang tepat untuk titik data. Untuk tugas regresi, random forest memprediksi dengan mencari rata-rata dari masing-masing pohon.

Ensemble learning adalah proses menggabungkan beberapa model machine learning untuk memecahkan masalah tertentu. Secara umum, ensemble learning digunakan untuk memperoleh hasil kinerja yang lebih baik dan mengurangi kemungkinan pemilihan model yang buruk. Ensemble learning yang digunakan pada random forest adalah bootstrap aggregating atau bagging. Langkah pertama dalam proses bagging adalah menghasilkan dataset bootstrap, kemudian training, lalu dilakukan agregasi untuk mendapatkan hasil akhir.



Gambar 8. Ilustrasi random forest

2.1.6 Metode Lasso regression

Lasso (Least Absolute Shrinkage and Selection Operator) regression merupakan teknik regularisasi L1 pada regresi, yang mana dilakukan penambahan penalti sebesar nilai absolut dari besarnya koefisien. Regularisasi dilakukan dengan tujuan untuk menghindari overfitting pada model. Pada data yang memiliki perbedaan variasi yang cukup jauh di antara data training dan data testnya, penambahan penalti yang diimplementasikan oleh regularisasi cukup berguna untuk memperoleh variansi yang lebih kecil pada data test.

Tujuan utama dari algoritma Lasso regression adalah meminimumkan fungsi berikut:

$$\sum_{i=1}^{n}\left(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

di mana,

- λ menotasikan besarnya penyusutan
- $\lambda = 0$ menunjukkan bahwa semua fitur akan dipertimbangkan pada model regresi, atau bisa dibilang ekuivalen dengan regresi linear tanpa regularisasi
- $\lambda = \infty$ menunjukkan bahwa tidak ada fitur yang dipertimbangkan. Hal ini berarti semakin nilai λ mendekati ∞, maka akan semakin banyak fitur yang dieliminasi dari model
- Bias akan bertambah seiring dengan bertambahnya nilai λ, sedangkan variansi akan berkurang seiring dengan bertambahnya nilai nilai λ

Pada acuan yang dijadikan referensi, digunakan nilai $\lambda = 0.012$. Oleh karena itu, dilakukan modifikasi pada nilai λ yang berbeda-beda untuk mengetahui pengaruh dan sejauh mana perbedaan nilai λ terhadap model yang dibuat.

2.1.7 Metode Ridge regression

Ridge regression merupakan teknik regularisasi L2 pada regresi. Sama seperti pada metode Lasso regression, metode Ridge regression memberikan penalti pada model untuk menghindari terjadinya overfitting. Yang membedakan Ridge regression dengan Lasso regression adalah pada Ridge regression melakukan penambahan penalti sebesar nilai kuadrat dari besarnya koefisien. Hal ini menyebabkan fungsi yang perlu diminumkan oleh Ridge regression dapat ditulis seperti berikut:

$$\sum_{i=1}^{n}\left(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

Ketika $\lambda = 0$, metode ini tidak akan jauh berbeda dengan OLS, yaitu regresi sederhana tanpa adanya penalti. Sedangkan ketika nilai λ cukup besar, mendekati tak hingga, koefisien dari Ridge regression akan dekat dengan nol, namun tidak akan pernah nol. Hal tersebut juga yang membedakan antara Ridge regression dengan Lasso regression, di mana Ridge akan tetap menjaga semua fitur, sedangkan Lasso regression cenderung menghapus fitur yang dianggap tidak relevan. Nilai λ yang digunakan pada acuan adalah sebesar 20000, yang kemudian akan dimodifikasi dengan melakukan variasi pada nilai λ yang berbeda-beda untuk melihat pengaruh terhadap model.

2.2 Metode Kembangan

2.2.1 K-Nearest Neighbour regression

Dalam statistik, algoritma k-nearest neighbor (k-NN) adalah metode pembelajaran terawasi non-parametrik yang dapat digunakan untuk melakukan klasifikasi dan regresi. Dalam kedua kasus, input terdiri dari *k* training sampel terdekat dalam kumpulan data. Outputnya tergantung pada apakah k-NN digunakan untuk klasifikasi atau regresi. Dalam regresi k-NN, outputnya adalah nilai properti untuk objek. Nilai ini merupakan rata-rata dari nilai k tetangga terdekat.

2.2.2 Neural Network

Neural Networks adalah bagian dari pembelajaran mesin dan merupakan inti dari algoritma deep learning. Nama dan strukturnya terinspirasi oleh otak manusia,

meniru cara neuron biologis memberi sinyal satu sama lain. Artificial Neural Networks (ANN) terdiri dari lapisan node, yang berisi lapisan input, satu atau lebih hidden layer, dan lapisan output. Setiap node, atau neuron buatan, terhubung ke yang lain dan memiliki bobot dan ambang yang terkait. Jika output dari setiap simpul individu berada di atas nilai ambang batas yang ditentukan, simpul tersebut diaktifkan, mengirimkan data ke lapisan jaringan berikutnya. Jika tidak, tidak ada data yang diteruskan ke lapisan jaringan berikutnya.

2.2.3 Polynomial Regression

Polinomial Regression merupakan bentuk dari analisis regresi pada hubungan antara variable bebas x dan variable bergantung y dengan dimodelkan sebagai polynomial berderajat n di x. Persamaan polynomial regression dapat ditulis seperti berikut.

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_n X^n + \epsilon$$

di mana,

Y : Variabel dependen

X : Variabel bebas

n : Derajat pada polynomial

β_i : Koefisien untuk X berderajat ke-i, $i = \{0,1, \dots, n\}$

ϵ : Error

Meskipun pada model ini mengijinkan untuk permasalahan hubungan non-linear antara Y dan X, Polynomial Regression masih dipertimbangkan sebagai regresi linear karena linear dalam koefisien regresi.

2.3 Evaluasi Model

Penulis melakukan evaluasi model menggunakan dua indikator, yaitu performa model dan efisiensi model. Untuk mengukur performa model, digunakan parameter Mean Square Error dan Mean Absolute Error. Sedangkan untuk mengukur efisiensi model, digunakan parameter waktu dan penggunaan memory. Berikut adalah penjelasan dari parameter-parameter yang akan digunakan.

2.3.1 Mean Square Error

Mean square error adalah rata-rata dari error yang dikuadratkan. Mean square error dapat dihitung menggunakan rumus berikut.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

dengan:

y_i : nilai aktual ke-i

$\hat{y}_i$: nilai prediksi ke-i

n : banyaknya data

2.3.2 Mean Absoute Error

Mean absolute error adalah rata-rata dari error yang dimutlakkan. Mean absolute error dapat dihitung menggunakan rumus berikut.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

BAB III
ANALISIS

3.1 Hasil

Pada bagian ini ditunjukkan perbandingan performa dan tingkat efisiensi dari setiap metode yang sudah disebutkan sebelumnya. Adapun untuk setiap metode yang ditulis pada tabel-tabel di bawah merupakan model dengan performa error terbaik di seriap metodenya. Sebagai contoh, pada metode Random Forest Regression, penulis melakukan training model dengan nilai max_depth yang berbeda-beda, yaitu: 4, 5, dan 6. Dikarenakan nilai $max\_depth = 6$ pada metode ini memberikan nilai MSE dan MAE yang lebih baik jika dibandingkan dengan lainnya, maka dipilihlah untuk kemudian akan dibandingkan dengan metode yang lainnya. Untuk lebih jelasnya mengenai proses pembandingan pada setiap metode akan dijelaskan di bagian pembahasan.

Algoritma dari setiap metode dijalankan pada *google colab* dan menggunakan GPU dengan spesifikasi sebagai berikut:

Parameter	Spesifikasi
GPU	Nvidia K80 / T4
GPU Memory	12GB / 16GB
GPU Memory Clock	0.82GHz / 1.59GHz
Performance	4.1 TFLOPS / 8.1 TFLOPS
Support Mixed Precision	No / Yes
GPU Release Year	2014 / 2018
No. CPU Cores	2
Available RAM	12GB (upgradable to 26.75GB)
Disk Space	358GB

Tabel 2. Spesifikasi environment yang digunakan dalam melakukan testing

3.1.1 Perbandingan performa

Terdapat 7 metode yang akan dibandingkan, yang terdiri dari 4 metode acuan yang telah dilakukan dimodifikasi dan 3 metode kembangan. Berikut hasil dari performa dari setiap metode.

Metode	MSE	MAE
Multiple Linear Regression	0.523 (+/- 0.00000517)	0.563 (+/- 0.00000153)
Random Forest Regression (max depth = 6)	0.374 (+/- 0.00000843)	0.456 (+/- 0.00000286)
Ridge Regression ($\alpha = 0.5$)	0.523 (+/- 0.00000517)	0.563 (+/- 0.00000153)
Lasso Regression ($\alpha = 0.012$)	0.561 (+/- 0.00000432)	0.590 (+/- 0.00000095)
KNN Regression ($k = 2$)	0.046 (non-cv) 0.065 (cv)	0.085 (non-cv) 0.111 (cv)
Polynomial Regression ($d = 3$)	0.228	0.326
Artificial Neural Network	0.299	0.416

Tabel 3. Perbandingan performa antara metode

3.1.2 Perbandingan efisiensi

Setiap proses training yang dilakukan pada masing-masing metode dilakukan pencatatan waktu eksekusi, yang meliputi *CPU times user, system,* dan *wall time*. Pencatatan waktu ekseskusi tidak dilakukan secara manual, melainkan menggunakan library $time$. *User time* adalah jumlah waktu CPU yang diambil di luar kernel, sedangkan *system time* diambil di luar kernel. Unruk *wall time* adalah waktu yang dibutuhkan yang tergantung pada dinding di luar komputer. Berikut adalah hasil nilai pencatatan waktu di setiap metodenya.

Metode	CPU times			Wall time
	User	System	Total	
Multiple Linear Regression	1.17 s	669 ms	1.84 s	1.08 s
Random Forest Regression (max depth = 6)	16 min 3 s	1.06 s	16 min 4 s	8 min 17 s
Ridge Regression ($\alpha = 0.5$)	777 ms	540 ms	1.32 s	748 ms
Lasso Regression ($\alpha = 0.012$)	4.76 s	642 ms	5.41 s	2.8 s
KNN Regression ($k = 2$)	24.8 s	29 ms	24.8 s	24.6 s
Polynomial Regression ($d = 3$)	12.8 s	1.38 s	14.2 s	11.1 s
Artificial Neural Network	8 min 58 s	43.4 s	9 min 42 s	8 min 23 s

Tabel 4. Perbandingan efisiensi waktu antara metode

Selain menghitung efisiensi waktu, setiap metode juga akan dibandingkan dari penggunaan memory pada setiap proses trainingnya. Pencatatan besar penggunaan memory dalam satuan MiB (Mebibyte) dilakukan menggunakan $memory\_profiler$

yang akan memberikan output berupa *Peak* dan *Increment* pada suatu algoritma yang diinginkan. Berikut tabel penggunaan memory di setiap metodenya.

Metode	Memory	
	Peak (MiB)	Increment (MiB)
Multiple Linear Regression	795.75	0.30
Random Forest Regression (max depth = 6)	783.38	18.67
Ridge Regression ($\alpha = 0.5$)	891.33	0.00
Lasso Regression ($\alpha = 0.012$)	890.65	0.00
KNN Regression ($k = 2$)	1018.34	0.02
Polynomial Regression ($d = 3$)	10379.01	0.00
Artificial Neural Network	9566.17	0.00

Tabel 5. Perbandingan efisiensi memory antara metode

3.2 Pembahasan

Berdasarkan hasil yang diperoleh pada subbab sebelumnya, terlihat bahwa model KNN regression mampu memberikan performa terbaik di antara metode yang lain. Nilai MSE dan MAE yang diperoleh dari metode KNN regression pada $k = 2$ relatif jauh lebih kecil jika dibandingkan dengan model yang menggunakan metode lainnya. Di sisi lain, Ridge regression hadir sebagai pemenang dalam perbandingan metode yang dianggap paling efisien. Metode Ridge regression hanya membutuhkan total waktu sebesar 1.32 s untuk menyelesaikan proses training, yang mana lebih cepat 0.52 s dengan total waktu yang dibutuhkan metode Multiple Linear Regression. Pada perbandingan efisiensi dari segi memory, Multiple Linear Regression membutuhkan memory paling sedikit dengan *Peak* sebesar 795.75 MiB dan Polynomial Regression untuk derajat tiga membutuhkan memory terbanyak sebesar 10379.01 MiB.

- Multiple Linear Regression

Model Multiple Linear Regression dapat dibilang sebagai model yang paling sederhana untuk menyelesaikan permasalahan regresi. Karena kesederhanaannya itulah metode ini tidak membutuhkan waktu eksekusi yang lama dan memory yang besar. Namun di sisi lain, efisiensi waktu dan memory pada metode ini harus dibayar dengan performa prediksi yang kurang optimal jika dibandingkan dengan beberapa metode lainnya. Proses training pada metode ini dilakukan secara *K-fold* dengan parameter $n\_split = 5$ dan $shuffle = True$. Hal ini dilakukan untuk menghindari terjadinya

overfitting pada model. Berikut hasil output dari metode ini yang menunjukkan nilai performa MAE dan MSE sekaligus efisiensi berdasarkan waktu dan memorinya.

```
============Multiple Linear Regression============
MSE scores : 0.523 (+/- 0.00000517)
MAE scores : 0.563 (+/- 0.00000153)

time: 1.5523107051849365
CPU times: user 1.17 s, sys: 669 ms, total: 1.84 s
Wall time: 1.08 s

peak memory: 795.75 MiB, increment: 0.30 MiB
==================================================
```

Gambar 9. Hasil multiple linear regression

- Random Forest Regression

Random forest regression adalah model yang mampu memberikan kinerja yang relatif cukup baik jika dibandingkan dengan yang lainnya namun dengan efisiensi terburuk. Penulis melakukan eksperimen menggunakan 3 max depth berbeda yaitu 4, 5, dan 6. Berikut adalah hasil keseluruhan eksperimen.

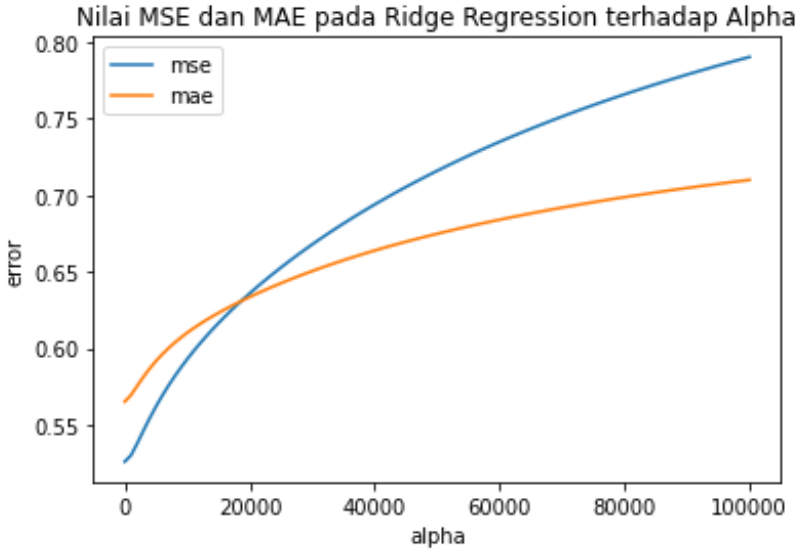
```
============Random Forest============
Random Forest (max_depth=4)
MSE scores : 0.475 (+/- 0.00000544)
MAE scores : 0.527 (+/- 0.00000107)

time: 358.19382214546204
CPU times: user 11min 8s, sys: 785 ms, total: 11min 9s
Wall time: 5min 46s

peak memory: 763.81 MiB, increment: 0.03 MiB
====================================
============Random Forest============
Random Forest (max_depth=5)
MSE scores : 0.427 (+/- 0.00000608)
MAE scores : 0.496 (+/- 0.00000165)

time: 1478.7438969612122
CPU times: user 13min 47s, sys: 905 ms, total: 13min 48s
Wall time: 7min 7s

peak memory: 764.30 MiB, increment: 0.04 MiB
====================================
============Random Forest============
Random Forest (max_depth=6)
MSE scores : 0.374 (+/- 0.00000843)
MAE scores : 0.456 (+/- 0.00000286)

time: 2832.116089820862
CPU times: user 16min 3s, sys: 1.06 s, total: 16min 4s
Wall time: 8min 17s

peak memory: 783.38 MiB, increment: 18.67 MiB
====================================
```

Gambar 10. Hasil random forest regression

Kinerja random forest regression menjadi lebih baik seiring bertambahnya max depth yang merupakan kedalaman maksimum dari pohon yang dibuat dalam random forest. Namun di sisi lain, meningkatnya max depth juga meningkatkan penggunaan resource sehingga tidak efisien secara waktu dan penggunaan memory.

- Ridge Regression

Ridge regression memberikan hasil kinerja yang mirip dengan Multiple Linear Regression. Dilakukan beberapa pengematan terhadap besar error yang dihasilkan oleh α yang divariasikan. Berikut ditunjukkan plot grafik besar MSE dan MAE oleh Ridge Regression terhadap nilai $\alpha = \{0, 1000, 2000, \dots, 100000\}$.



Gambar 11. Nilai MSE dan MAE terhadap alpha pada ridge regression

Dari grafik terlihat bahwa untuk penerapan pada dataset ini, nilai MSE dan MAE akan berkurang seiring dengan berkurangnya nilai α. Selanjutnya, untuk tingkat efisiensi algoritma pada metode Ridge Regression akan diuji pada beberapa nilai α.

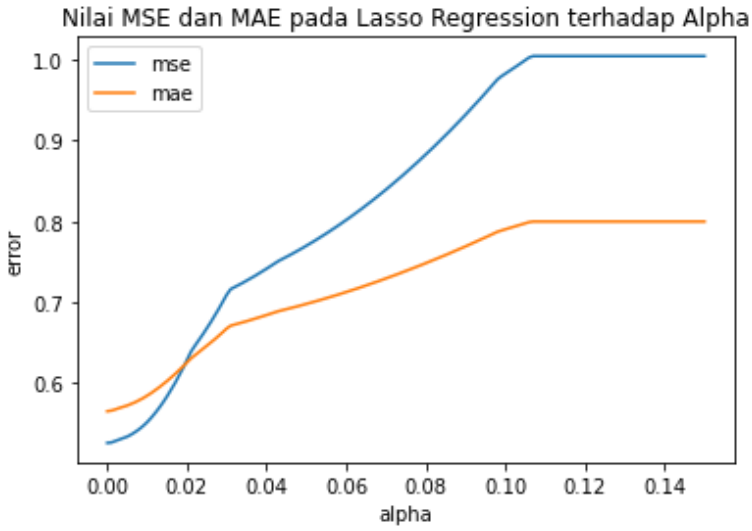
```
============Ridge Regression============
Ridge (alpha=0.5)
MSE scores : 0.523 (+/- 0.00000517)
MAE scores : 0.563 (+/- 0.00000153)

time: 0.6895694732666016
CPU times: user 777 ms, sys: 540 ms, total: 1.32 s
Wall time: 748 ms

peak memory: 891.33 MiB, increment: 0.00 MiB
========================================
============Ridge Regression============
Ridge (alpha=500)
MSE scores : 0.524 (+/- 0.00000513)
MAE scores : 0.565 (+/- 0.00000148)

time: 0.7368865013122559
CPU times: user 784 ms, sys: 518 ms, total: 1.3 s
Wall time: 720 ms

peak memory: 891.34 MiB, increment: 0.01 MiB
========================================
============Ridge Regression============
Ridge (alpha=10000)
MSE scores : 0.591 (+/- 0.00000488)
MAE scores : 0.609 (+/- 0.00000097)

time: 0.7250382900238037
CPU times: user 766 ms, sys: 546 ms, total: 1.31 s
Wall time: 709 ms

peak memory: 891.34 MiB, increment: 0.00 MiB
========================================
```

Gambar 12. Hasil ridge regression

Terlihat bahwa untuk penggunaan memory dan total waktu yang dibutuhkan pada α yang divariasikan cenderung cukup stabil. Sebagai catatan tambahan, pada metode ini pembagian dataset raining dan test juga dilakukan secara *K-fold* dengan parameter $n\_split = 5$ dan $shuffle = True$.

- Lasso Regression

Lasso Regression pada $\alpha = 0.012$ memiliki nilai performa yang tidak lebih baik dari metode Ridge Regression dan Multiple Linear Regression, meskipun jika dilihat secara selisih tidak berbeda terlalu jauh. Cara yang sama seperti metode sebelumnya dilakukan untuk melihat perubahan nilai MSE dan MAE terhadap α yang divariasikan. Berikut ditunjukkan plot grafik besar MSE dan MAE oleh Lasso Regression terhadap nilai $\alpha = \{0, 0.001, 0.002, \dots, 0.15\}$.



Gambar 13. Nilai MSE dan MAE terhadap alpha pada lasso regression

Lasso regression menerapkan penalti pada sebuah regresi sebagaimana yang dilakukan oleh Ridge Regression. Hanya saja, pada Lasso Regression ketika nilai α bertambah besar, beberapa fitur akan dieliminasi oleh model. Hal ini dapat menjelaskan perilaku nilai MSE dan MAE yang cenderung stabil ketika telah menyentuh suatu nilai α pada grafik di atas. Berbeda dengan Ridge Regression, ketika diaplikasikan pada dataset yang sedang digunakan, metode Lasso Regression jauh lebih sensitif terhadap perubahan nilai α, namun lebih stabil ketika nilai α telah mencapai suatu angka tertentu. Selanjutnya, untuk tingkat efisiensi algoritma pada metode Lasso Regression akan diuji pada beberapa nilai α.

```
============Lasso Regression============
Lasso (alpha=0.012)
MSE scores : 0.561 (+/- 0.00000432)
MAE scores : 0.590 (+/- 0.00000095)

time: 3.643394947052002
CPU times: user 4.76 s, sys: 642 ms, total: 5.41 s
Wall time: 2.8 s

peak memory: 890.65 MiB, increment: 0.00 MiB
========================================
============Lasso Regression============
Lasso (alpha=0.05)
MSE scores : 0.768 (+/- 0.00000323)
MAE scores : 0.697 (+/- 0.00000081)

time: 0.8388485908508301
CPU times: user 1.02 s, sys: 541 ms, total: 1.56 s
Wall time: 863 ms

peak memory: 890.69 MiB, increment: 0.00 MiB
========================================
============Lasso Regression============
Lasso (alpha=0.15)
MSE scores : 1.003 (+/- 0.00000175)
MAE scores : 0.799 (+/- 0.00000060)

time: 0.7860667705535889
CPU times: user 883 ms, sys: 529 ms, total: 1.41 s
Wall time: 782 ms

peak memory: 890.69 MiB, increment: 0.00 MiB
========================================
```

Gambar 14. Hasil lasso regression

Dilihat dari segi efisiensi, Lasso Regression terlihat tidak memiliki masalah kestabilan pada penggunaan memory. Namun, Lasso Regression cukup membutuhkan memakan waktu yang lebih lama untuk nilai α yang lebih kecil. Pada proses training ini, pembagian dataset training dan test juga dilakukan secara *K-fold* dengan parameter $n\_split = 5$ dan $shuffle = True$.

- KNN Regression

KNN regression memberikan performa yang cukup memuaskan ketika digunakan pada dataset yang sudah dirujuk di awal. Nilai MSE yang diperoleh sebesar 0.046 (non-cv) dan 0.065 (cv) untuk $k = 2$ menjadikan model ini memiliki gap performa yang cukup jauh dibandingkan model dengan metode lainnya, yang mana metode lainya bahkan nilai MSE dan MAE belum mampu menyentuh angka 0.1. Penentuan nilai $k = 2$ diperoleh dari percobaan input parameter $k = \{1,2,..,5\}$ dan dilakukan *cross-validation* menggunakan bantuan library *GridSearchCV*.

```
#Model Tuning (learning best n_neighbors hyperparameter)
knn_params={'n_neighbors' : np.arange(1,5,1)}

knn=KNeighborsRegressor()
knn_cv_model=GridSearchCV(knn, knn_params, cv=5)

knn_cv_model.fit(X_train,y_train)
best_parameter_k = knn_cv_model.best_params_["n_neighbors"]
print(f"Parameter terbaik untuk k [k=1..5]: {best_parameter_k} ")
```

```
Parameter terbaik untuk k [k=1..5]: 2
```

Gambar 15. Menentukan nilai k yang optimal

Selanjutnya diberikan hasil penggunaan memory dan waktu yang dibutuhkan untuk nilai $k = 1$ dan nilai $k = 2$ untuk melihat pengaruh parameter k terhadap kebutuhan *resources*.

```
============K-Nearest Neighbour Regression============
k = 1
MSE value: 0.050938845287540564
MSE values with applying Cross Validation: 0.07130863275706992
MAE value: 0.07878279133883793
MAE values with applying Cross Validation: 0.10075477535718233

time: 18.85040783882141
CPU times: user 18.2 s, sys: 28.6 ms, total: 18.3 s
Wall time: 18.2 s

peak memory: 1018.32 MiB, increment: 0.04 MiB
=======================================================
============K-Nearest Neighbour Regression============
k = 2
MSE value: 0.04583043435645167
MSE values with applying Cross Validation: 0.0651640524226525
MAE value: 0.08488054762586045
MAE values with applying Cross Validation: 0.11141321877425024

time: 24.64142155647278
CPU times: user 24.8 s, sys: 29 ms, total: 24.8 s
Wall time: 24.6 s

peak memory: 1018.34 MiB, increment: 0.02 MiB
=======================================================
```

Gambar 16. Hasil KNN regression

Penggunaan memory yang dibutuhkan dari kedua nilai k yang berbeda cukup stabil, namun pada $k = 2$ terlihat membutuhkan waktu eksekusi yang lebih lama dibandingkan dengan $k = 1$. Pada proses training dengan metode ini pembagian dataset training dan test dilakukan secara dua cara, dengan *cross validation* (cv) dan tanpa *cross validation* (non cv). Pada pembagian dataset menggunakan *cross validation* digunakan nilai $cv = 2$.

- Polynomial Regression

Polynomial regression memberikan hasil yang baik namun dengan penggunaan resource yang cukup besar pula. Penulis melakukan eksperimen dengan mencoba memvariasikan parameter derajat tertinggi dari polinom yaitu 2 dan 3. Hasilnya adalah sebagai berikut.

```
============Polynomial Linear Regression============
degree=2
MSE scores : 0.356
MAE scores : 0.434
====================================================

time: 1.669513463973999
CPU times: user 1.83 s, sys: 96.7 ms, total: 1.93 s
Wall time: 1.48 s
(0.35648865982667727, 0.4343499702460889)

peak memory: 10480.14 MiB, increment: 0.36 MiB

============Polynomial Linear Regression============
degree=3
MSE scores : 0.228
MAE scores : 0.326
====================================================

time: 11.73393440246582
CPU times: user 12.8 s, sys: 1.38 s, total: 14.2 s
Wall time: 11.1 s
(0.2279445527348753, 0.3257249670742581)

peak memory: 10379.01 MiB, increment: 0.00 MiB
```

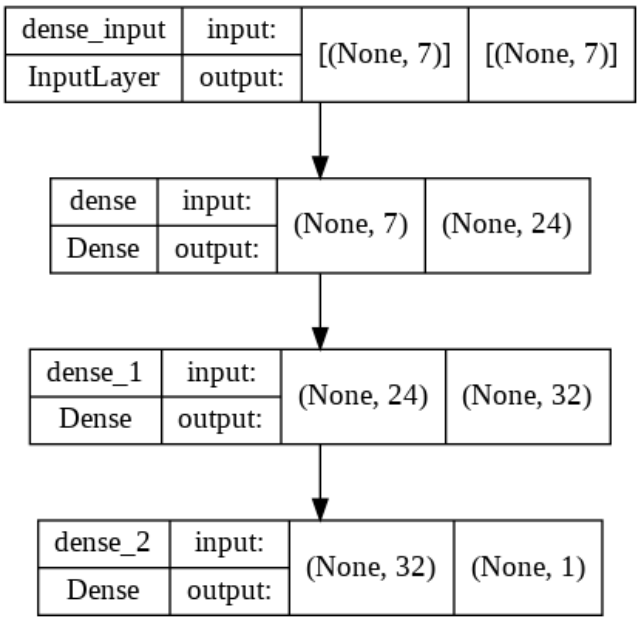
Gambar 17. Hasil polynomial regression

Dari hasil tersebut, terlihat bahwa performa terbaik diberikan oleh polinom dengan derajat 3. Secara memory, polinom berderajat 2 dan 3 juga tidak jauh berbeda, bahkan polinom berderajat 3 justru sedikit lebih kecil dari derajat 2. Namun, jika dilihat dari efisiensi waktu, polinom berderajat 3 memakan waktu jauh lebih lama dibandingkan derajat 2.

- Artificial Neural Network

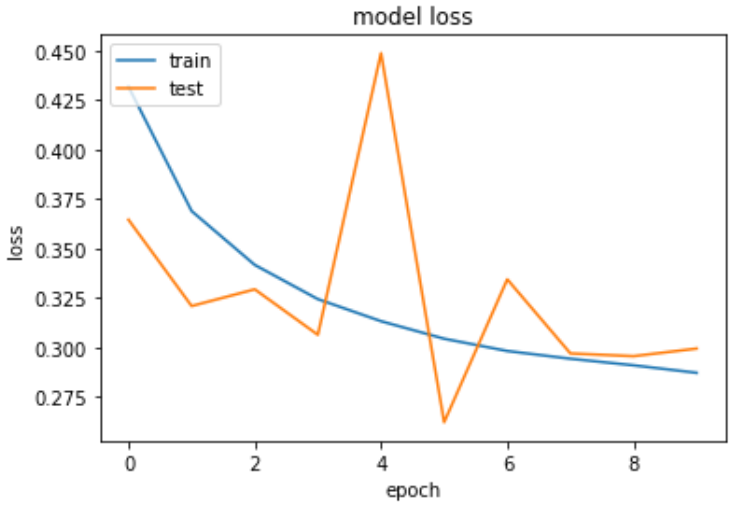
Artificial neural network adalah salah satu model paling kompleks yang penulis coba. Karena kompleksitasnya, model ini sangat tidak efisien secara memory maupun waktu jika dibandingkan dengan model-model lainnya. Selain itu, ternyata model ini juga

tidak berhasil menghasilkan performa terbaik. Model yang penulis gunakan menggunakan tiga layer, dengan tiap layernya memiliki jumlah input dan output seperti berikut.

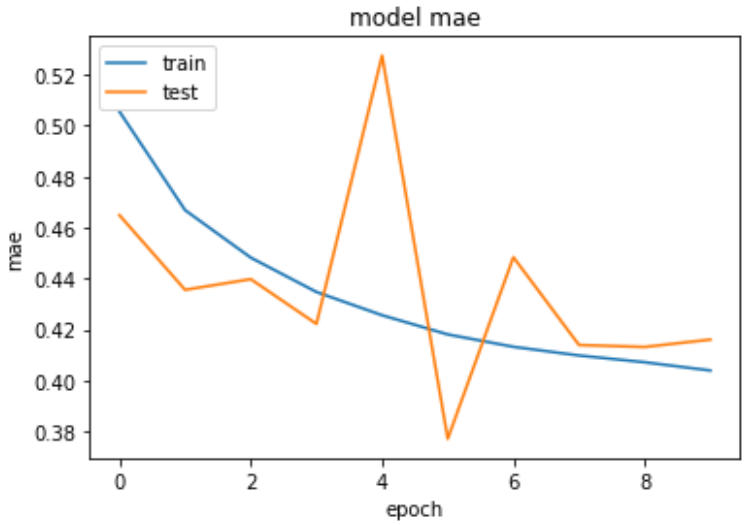


Gambar 18. Model neural network yang digunakan

Dengan $activation = relu$, $loss = mse$, $epoch = 10$, dan $batch\_size = 50$, diperoleh nilai MAE dan MSE di setiap *epoch*-nya sebagai berikut.



Gambar 19. Grafik MSE setiap epoch



Gambar 20. Grafik MAE setiap epoch

Nilai MSE dan MAE sempat melonjak naik pada *epoch* ke-4 dan kemudian turun pada *epoch* setelahnya. Model menggunakan metode Neural Network untuk dataset ini dirasa kurang optimal untuk memberikan performa yang baik. Proses training pada metode ini juga membutuhkan waktu yang relatif lama dan memory yang cukup besar. Berikut informasi penggunaan waktu dan memory yang dibutuhkan.

```
CPU times: user 8min 58s, sys: 43.4 s, total: 9min 42s
Wall time: 8min 23s

peak memory: 9566.17 MiB, increment: 0.00 MiB
=====================================
```

Gambar 21. Hasil efisiensi waktu dan memory pada model neural network

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan percobaan yang dilakukan dan hasil yang diperoleh, kita dapat menggunakan beberapa macam regresi untuk melakukan prediksi suhu magnet permanen pada motor listrik. Penulis mencoba menerapkan 7 model untuk problem ini, yaitu multiple linear regression, random forest regression, ridge regression, lasso regression, k-nearest neighbour (KNN) regression, artificial neural network, dan polynomial regeression.

Dari 7 model yang dicoba, model KNN regression adalah model yang memberikan performa terbaik, sedangkan model ridge regression adalah model yang memberikan efisiensi terbaik baik dari waktu maupun memory. Secara keseluruhan, model KNN regression adalah model terbaik dan paling cocok untuk problem ini karena dapat memberikan hasil performa terbaik dan secara efisiensi waktu maupun memory pun tidak buruk meskipun bukan yang terbaik.

4.2 Saran

Saran dari penulis untuk penelitian selanjutnya adalah dapat dilakukan pengujian lebih lanjut dengan menggunakan dataset pada laporan ini sebagai benchmark. Penggunaan metode yang sudah dibahas oleh penulis pada laporan ini dapat digunakan kembali pada dataset yang lebih terupdate daripada yang penulis gunakan (penulis masih menggunakan dataset *version 2* dan sekarang telah tersedia dataset *version 3* ketika penulis melakukan penelitian ini). Hal ini dilakukan untuk mengetahui sebaik apa metode yang dibahas dalam menghadapi jumlah data yang semakin besar dan terus diperbarui. Selain itu, dapat dilakukan penambahan langkah pada pre-processing yang belum dilakukan oleh penulis (misal: PCA) dan melakukan eksplorasi dengan menggunakan metode-metode lain yang belum disebutkan untuk melihat perbedaan performa dan efisiensi dari metode-metode yang digunakan.

DAFTAR PUSTAKA

Asarkar, P. (2020). *PMSM Rotor Temperature Detection*. Diambil kembali dari Kaggle: https://www.kaggle.com/code/pratikasarkar/pmsm-rotor-temperature-detection/notebook

IBM Cloud Education. (2020). *What is Supervised Learning?* Diambil kembali dari IBM: https://www.ibm.com/cloud/learn/supervised-learning

Kirchgässner, W., Wallscheid, O., & Böcker, J. (2021). Estimating Electric Motor Temperatures With Deep Residual Machine Learning. *IEEE Transactions on Power Electronics*, 7480-7488.

Kirgsn. (2018). *Electric Motor Temperature*. Diambil kembali dari Kaggle: https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature

Kuruma, V. (2019). *Regression in Machine Learning: What it is and Examples of Different Models*. Diambil kembali dari Built In: https://builtin.com/data-science/regression-machine-learning

Nagpal, A. (2017). *L1 and L2 Regularization Methods*. Diambil kembali dari Medium: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

Pedregosa, et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825-2830.

PennState Eberly College of Science. (2018). *Polynomial Regression*. Diambil kembali dari PennState Eberly College of Science: https://online.stat.psu.edu/stat462/node/158/

Singh, A. (2018). *A Practical Introduction to K-Nearest Neighbors Algorithm for Regression (with Python code)*. Diambil kembali dari Analytics Vidhya: https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

LAMPIRAN

File-file yang digunakan pada laporan ini dapat ditemukan pada folder yang terletak di file zip sama dengan laporan ini, yang terdiri dari file kode berformat .ipynb, file html yang berisi kode dan output yang telah dijalankan, serta dataset yang digunakan.

1) 10118030-10118104-Tubes2022.ipynb
2) 10118030-10118104-Tubes2022.html
3) pmsm_temperature_data.csv