LAPORAN TUGAS PERMODELAN STATISTIK TERAPAN

MATA KULIAH STATISTIKA TERAPAN

“Ridge & Lasso Regression Pada Dataset IPM 2016-2018”

Dosen Pengampu: Fitrah Maharani Humaira M.Kom

Disusun Oleh:

AULYA BERLYANA AGUSTIN

NRP. 3324600051

SAINS DATA TERAPAN B

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER

PROGRAM STUDI SAINS DATA TERAPAN

PRAKTIKUM RIDGE & LASSO REGRESSION

“IPM Jatim 2016- 2018”

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Indeks Pembangunan Manusia (IPM) di Jawa Timur merupakan indikator penting yang mencerminkan kualitas hidup dan kesejahteraan masyarakat di provinsi tersebut. Peningkatan mutu hidup manusia telah menjadi perhatian utama dalam proses pembangunan suatu provinsi. Salah satu metode yang digunakan untuk mengukur mutu hidup adalah Indeks Pembangunan Manusia (IPM), yang meliputi aspek-aspek seperti kesehatan, pendidikan, dan standar hidup. IPM mencerminkan kemajuan suatu wilayah dalam memenuhi kebutuhan dasar penduduknya. Di Provinsi Jawa Timur seperti halnya di banyak daerah di Indonesia, peningkatan IPM dianggap sebagai bagian penting dari upaya pembangunan yang berkelanjutan.

Dengan memanfaatkan analisis seperti Ridge dan Lasso Regression dapat menjadi alternatif Solusi untuk memberikan pemahaman yang berharga terhadap factor-faktor yang mempengaruhi IPM di Provinsi Jawa Timur. Melalui pendekatan ini diharapkan dapat mendukung perancangan kebijakan pemerintah yang lebih efektif untuk meningkatkan mutu hidup penduduk di Jawa Timur

1.2 Tujuan

Tujuan dari Pemrosesan data ini sebagai bentuk implementasi dari penerapan metode Ridge Regression dan Lasso Regression untuk mengatasi multikolinieritas serta melakukan komparasi metode antara OLS, Ridge, dan Lasso berdasarkan metriks-metriks evaluasi yang ditentukan, memilih persamaan regresi terbaik dan menggunakannya sebagai model prediktif. Selain itu pemrosesan ini bertujuan untuk mengetahui faktor apa saja yang berpengaruh secara signifikan terhadap IPM di Provinsi Jawa Timur pada tahun 2016 hingga 2018 dan juga mengetahui bagaimana cara kerja model regresi linier, regresi Ridge, regresi Lasso pada tahun 2016 hingga 2018

1.2 Rumusan Masalah

1. Faktor apa saja yang mempengaruhi IPM pada tahun 2016-2018 secara signifikan di Provinsi Jawa Timur?
2. Mengetahui cara kerja model regresi linear dalam memprediksi IPM Provinsi Jawa Timur berdasarkan data tahun 2016-2018.
3. Mengetahui cara kerja model regresi Ridge dalam memprediksi IPM Provinsi Jawa Timur berdasarkan data tahun 2016-2018
4. 4. Mengetahui cara kerja model regresi Lasso dalam memprediksi IPM Provinsi Jawa Timur berdasarkan data tahun 2016 hingga 2018

BAB II. TINJAUAN PUSTAKA

2.1 Dasar Teori

Ridge Regression atau Regresi Ridge merupakan metode pendugaan koefisien regresi yang diperoleh melalui penambahan konstanta bias pada diagonal XX'. Meskipun metode ini menghasilkan penduga koefisien regresi yang berbias, penduga ini bisa mendekati nilai parameter yang sebenarnya. Regresi Ridge mengatasi masalah multikolinieritas dengan menentukan penduga yang bias tetapi mempunyai varian yang lebih kecil dari varian penduga regresi linier berganda (ordinary least square/OLS). Dalam metode ini dilakukan penyetelan model (model tuning) yang digunakan untuk menganalisis data apapun yang mengalami multikolinearitas. Metode ini melakukan regularisasi L2. Regresi Ridge tidak memberlakukan pemilihan fitur, regresi ridge hanya mengurangi bobot fitur kearah nol tetapi tidak pernah menjadi nol. Dalam Regresi Ridge, langkah pertama adalah menstandardisasi variabel (baik dependen maupun independen) dengan mengurangkan rata-ratanya dan membaginya dengan standar deviasinya. Hal ini menyebabkan tantangan dalam notasi karena bagaimanapun harus ditunjukkan apakah variabel dalam formula tertentu distandarisasi atau tidak. Sejauh menyangkut standarisasi, semua perhitungan Regresi Ridge didasarkan pada variabel standar. Ketika koefisien regresi akhir ditampilkan, maka disesuaikan kembali ke skala aslinya. Regresi Ridge melakukan 'regularisasi L2', yaitu menambahkan faktor jumlah kuadrat koefisien dalam tujuan optimasi.

Least Absolute Shrinkage Selection and Operator (LASSO) atau Regresi Lasso menyusutkan koefisien regresi dari variabel bebas yang memiliki korelasi tinggi menjadi tepat pada nol atau mendekati nol. Tujuan dari Regresi Lasso adalah untuk mendapatkan subset dari prediktor yang meminimalkan kesalahan prediksi untuk variabel respon kuantitatif. Lasso melakukan ini dengan memberlakukan batasan pada parameter model yang menyebabkan koefisien regresi untuk beberapa variabel menyusut menuju nol. regularisasi Lasso dengan mudah diperluas ke model statistik lainnya termasuk model linier umum, persamaan perkiraan umum dan model proporsional Hazard. Kemampuan Lasso untuk melakukan seleksi subset bergantung pada bentuk kendala dan memiliki berbagai interpretasi termasuk dalam hal geometri, statistik Bayesian dan analisis cembung. Regresi Lasso melakukan regularisasi L1, yaitu menambahkan faktor penjumlahan nilai mutlak koefisien dalam tujuan optimasi.

2.2 Indeks Pembagunan Manusia

IPM adalah rata rata geometris dari indeks normalisasi untuk masing-masing dari tiga dimensi. Tiga dimensi tersebut adalah dimensi kesehatan, dimensi pendidikan, dan dimensi standar hidup. Dimensi kesehatan dinilai oleh harapan hidup saat lahir. Dimensi pendidikan diukur dengan rata-rata masa sekolah untuk orang dewasa berusia 25 tahun dan lebih, dan, tahun sekolah yang diharapkan untuk anak-anak sekolah memasuki usia tersebut. Dimensi standar hidup diukur dengan pendapatan nasional kotor per kapita (BPS, 2017).

2.3 Faktor yang Mempengaruhi IPM

2.3.1 Angka Kematian Bayi

Angka kematian bayi (AKB) adalah banyaknya kematian bayi pada usia di bawah 1 tahun per 1.000 kelahiran hidup. AKB menjadi salah satu tolak ukur untuk menilai sejauh mana ketercapaian kesejahteraan rakyat sebagai hasil dari pelaksanaan pembangunan di bidang kesehatan. Pembangunan kesehatan ini dapat tercapai dengan meningkatkan derajat kesehatan masyarakat Indonesia. Salah satu derajat kesehatan yang belum dicapai oleh pemerintah adalah banyaknya jumlah kematian bayi. (Badan Pusat Statistik. Sistem Informasi Rujukan Statistik. Jakarta. 2015. [Online]. Available: https://sirusa.bps.go.id/index.php?r=indikator/view&id.)

2.3.2 Keluhan Kesehatan dan Sarana Kesehatan

Keluhan kesehatan adalah kondisi di mana seseorang mengalami gangguan kesehatan, baik fisik maupun mental. Menurut Badan Pusat Statistik (BPS) dan Kementerian Kesehatan RI, keluhan kesehatan mencakup berbagai masalah, seperti penyakit akut, penyakit kronis, kecelakaan, atau kondisi psikologis yang dapat memengaruhi aktivitas sehari-hari. Keluhan ini dapat disebabkan oleh berbagai faktor, baik internal (seperti genetik atau kondisi kesehatan sebelumnya) maupun eksternal (seperti lingkungan atau pola hidup). Keluhan kesehatan yang signifikan dapat menghambat upaya peningkatan kesejahteraan individu dan masyarakat secara keseluruhan (https://media.neliti.com/media/publications/446468-none-f3f5f518.pdf)

2.3.3 Angka Melek Huruf

Angka Melek Huruf (AMH) adalah indikator yang menunjukkan kemampuan penduduk untuk membaca dan menulis, biasanya diukur pada populasi berusia 15 tahun ke atas. AMH tidak hanya mencerminkan keterampilan literasi dasar, tetapi juga berfungsi sebagai indikator penting dalam menilai kemajuan pendidikan dan pengembangan sumber daya manusia di suatu negara.

2.3.4 APS-SMA

Angka Partisipasi Sekolah untuk jenjang SMA (APS-SMA) adalah indikator yang menunjukkan persentase siswa yang terdaftar di sekolah menengah atas (SMA) dibandingkan dengan jumlah anak yang seharusnya bersekolah pada usia tersebut. APS-SMA mencerminkan akses dan keterlibatan masyarakat dalam pendidikan menengah, yang merupakan langkah penting dalam mencapai pendidikan yang berkualitas dan meningkatkan kualitas sumber daya manusia.

2.3.5 Persentase Miskin

Persentase miskin adalah ukuran yang menunjukkan proporsi penduduk yang hidup di bawah garis kemiskinan dalam suatu wilayah. Garis kemiskinan biasanya ditentukan berdasarkan kebutuhan dasar, seperti makanan, pakaian, tempat tinggal, dan akses terhadap layanan kesehatan serta pendidikan. Penduduk yang dianggap miskin adalah mereka yang pengeluarannya tidak cukup untuk memenuhi kebutuhan minimum tersebut.

2.3.6 PDRB

Produk Domestik Regional Bruto (PDRB) adalah indikator ekonomi yang menggambarkan total nilai tambah yang dihasilkan oleh seluruh unit usaha dalam suatu wilayah selama periode tertentu, biasanya satu tahun. PDRB mencakup nilai barang dan jasa akhir yang dihasilkan di dalam batas wilayah geografis tertentu, baik oleh penduduk lokal maupun oleh perusahaan asing yang beroperasi di wilayah tersebut.

2.3.7 Pertumbuhan Ekonomi

Pertumbuhan Ekonomi adalah proses peningkatan kapasitas produksi suatu perekonomian yang tercermin dalam kenaikan output nasional atau pendapatan per kapita selama periode tertentu. Pertumbuhan ekonomi diukur dengan menggunakan indikator seperti Produk Domestik Bruto (PDB) atau Produk Domestik Regional Bruto (PDRB). Secara umum, pertumbuhan ekonomi mencerminkan perubahan positif dalam kondisi ekonomi suatu negara, yang dapat dilihat dari peningkatan jumlah barang dan jasa yang diproduksi. (jbie_patric)

2.3.8 Pengangguran Terbuka

Pengangguran Terbuka adalah kondisi di mana individu yang tergolong dalam angkatan kerja tidak memiliki pekerjaan dan secara aktif mencari pekerjaan. Menurut definisi yang diacu dari berbagai jurnal, pengangguran terbuka mencakup orang-orang yang berusia 15 tahun ke atas yang ingin bekerja, baik yang baru pertama kali mencari pekerjaan maupun mereka yang sudah memiliki pengalaman kerja sebelumnya tetapi belum mendapatkan pekerjaan baru.

BAB III. Percobaan Dataset 2016

3.1 Percobaan Dataset IPM 2016.csv

3.1.1 Install Library

```
import pandas as pd
import numpy as np
import itertools
import matplotlib.pyplot as plt
import time
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import LabelEncoder,scale,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
```

3.1.2 Membaca Dataset

Pada percobaan ini menerapkan model OLS menggunakan data IPM JATIM 2016

```
df = pd.read_excel('IPM Jatim 2016.xlsx')
df.head()
```

	Daerah	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka	IPM
0	Pacitan	19.56	58.94	1	91.54	68.71	15.49	5.21	5.21	1.00	65.74
1	Ponorogo	21.55	60.14	5	89.74	83.53	11.75	5.29	5.29	3.94	68.93
2	Trenggalek	18.01	57.45	1	94.32	70.44	13.24	5.00	5.00	3.07	67.78
3	Tulungagung	18.67	59.07	9	96.88	76.24	8.23	5.02	5.02	3.60	70.82
4	Blitar	20.49	57.72	7	93.56	67.06	9.88	5.08	5.08	2.92	68.88

Variabel	Simbol
Angka Kematian Bayi	X1
Keluhan Kesehatan	X2
Sarana Kesehatan	X3
Angka Melek Huruf	X4
APS-SMA	X5
Persentase Miskin	X6
PDRB	X7
Pertumbuhan Ekonomi	X8
Pengangguran Terbuka	X9

Kolom Daerah dihapus karena kolom daerah tidak berisikan tipe data numerik sehingga tidak diperlukan dalam perhitungan.

3.1.3 Matriks Korelasi

Menentukan matriks korelasi dari semua variabel atau fitur dari dataset yang digunakan

```
df_clean = df.dropna().drop(['Daerah'], axis=1)
print("Dimension of modified data :", df_clean.shape)
```

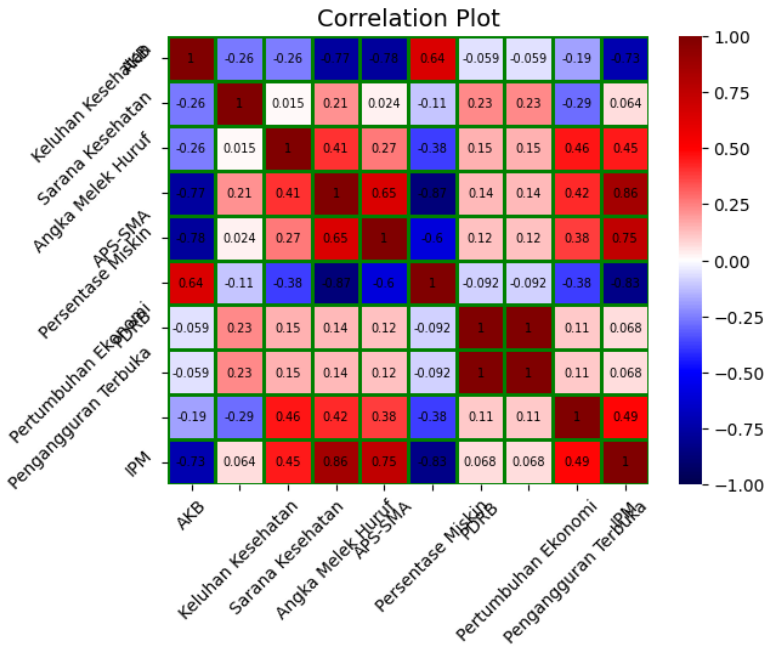
```
Dimension of the original data: (38, 11)
Dimension of modified data : (48, 10)
```

```
kw = df_clean.copy()
```

```
corr_matrix = kw.corr()
corr_matrix
```

	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka	IPM
AKB	1.000000	-0.259591	-0.257464	-0.772463	-0.776740	0.640973	-0.059058	-0.059058	-0.185607	-0.725439
Keluhan Kesehatan	0.259591	1.000000	0.015062	0.214236	0.024450	-0.111547	0.230477	0.230477	0.285351	0.063606
Sarana Kesehatan	0.257464	0.015062	1.000000	0.405946	0.265254	-0.379024	0.153567	0.153567	0.456344	0.448965
Angka Melek Huruf	-0.772463	0.214236	0.405946	1.000000	0.647112	-0.867472	0.135767	0.135767	0.415984	0.857827
APS-SMA	-0.776740	0.024450	0.265254	0.647112	1.000000	-0.596427	0.122694	0.122694	0.377336	0.748227
Persentase Miskin	0.640973	-0.111547	-0.379024	-0.867472	-0.596427	1.000000	-0.091843	-0.091843	-0.380596	-0.829183
PDRB	0.059058	0.230477	0.153567	0.135767	0.122694	-0.091843	1.000000	1.000000	0.107270	0.068316
Pertumbuhan Ekonomi	-0.059058	0.230477	0.153567	0.135767	0.122694	-0.091843	1.000000	1.000000	0.107270	0.068316
Pengangguran Terbuka	-0.185607	-0.285351	0.456344	0.415984	0.377336	-0.380596	0.107270	0.107270	1.000000	0.487293
IPM	-0.725439	0.063606	0.448965	0.857827	0.748227	-0.829183	0.068316	0.068316	0.487293	1.000000

```
sns.heatmap(corr_matrix,
            cmap = 'seismic',
            linewidth = 0.75,
            linecolor = 'green',
            cbar = True,
            vmin = -1,
            vmax = 1,
            annot = True,
            annot_kws = {'size': 7,'color':'black'})
plt.tick_params(labelsize = 10, rotation = 45)
plt.title('Correlation Plot',size = 14)
```



Dari output yang dihasilkan dapat disimpulkan bahwa nilai korelasi yang cenderung warna merah berarti cukup tinggi(korelasi positif). Warna biru juga mempunyai korelasi yang cukup tinggi (korelasi negatif). Nilai korelasi diatas 0,5 merupakan nilai korelasi yang paling bagus. Akan tetapi apabila mendekati 1 harus dicek ulang hubungan antara variabel independent karena takutnya apabila sudah mendekati 1 akan terjasi multikolinearitas.

3.1.4 Inisiasi Variabel Responden & Prediktor

```
y = df_clean['IPM']
X = df_clean.drop(['IPM'], axis=1)
X.head()
```

✓ 0.0s

	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka
0	19.56	58.94	1	91.54	68.71	15.49	5.21	5.21	1.00
1	21.55	60.14	5	89.74	83.53	11.75	5.29	5.29	3.94
2	18.01	57.45	1	94.32	70.44	13.24	5.00	5.00	3.07
3	18.67	59.07	9	96.88	76.24	8.23	5.02	5.02	3.60
4	20.49	57.72	7	93.56	67.06	9.88	5.08	5.08	2.92

Dari percobaan diatas didapatkan bahwa Variabel Responden (Y) adalah kolom IPM dan Variabel Prediktor (X) adalah kolom AKB, Keluhan Kesehatan, Angka Melek Huruf, APS-SMA, Persentase Miskin, PDRB, Pertumbuhan Ekonomi, Pengangguran Terbuka. Dari dataset ini dapat dianalisis seberapa pengaruhnya variabel prediktor(X) terhadap variabel responden(Y).

3.1.5 Regresi Linier

```
def processsubset(featureset):
    model = sm.OLS(y,X[list(featureset)])
    regr = model.fit()
    RSS = ((regr.predict(X[list(featureset)])-y)**2).sum()
    return {"model":regr,"RSS":RSS}
```

✓ 0.0s

Fungsi ini berguna untuk mengevaluasi kombinasi variabel prediktor dalam regresi linear. Nilai RSS yang lebih kecil menunjukkan model yang lebih baik dalam memprediksi data. Hasil dari variabel model dapat digunakan untuk analisis lebih lanjut seperti uji signifikansi, R-squared, dan interpretasi koefisien regresi.

```
def getbest(k):
    tic = time.time()
    results = []
    for combo in itertools.combinations(X.columns,k):
        results.append(processsubset(combo))
    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print("Processed", models.shape[0], k, "predictors in", toc-tic, "seconds")

    return best_model
```

Fungsi getbest(k) berguna untuk mencari kombinasi fitur terbaik dari ukuran k berdasarkan RSS terkecil. Digunakan dalam metode seleksi fitur untuk menentukan kombinasi optimal dalam regresi linear. Hasilnya bisa dianalisis lebih lanjut untuk interpretasi koefisien dan signifikansi statistik.

```
model_best = pd.DataFrame(columns = ["RSS","model"])

tic = time.time()
for i in range(1,10):
    model_best.loc[i] = getbest(i)
toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

✓ 1.9s

```
Processed 9 1 predictors in 0.07582807540893555 seconds
Processed 36 2 predictors in 0.15511178970336914 seconds
Processed 84 3 predictors in 0.3127460479736328 seconds
Processed 126 4 predictors in 0.5693204402923584 seconds
Processed 126 5 predictors in 0.3854360580444336 seconds
Processed 84 6 predictors in 0.2459700107574463 seconds
Processed 36 7 predictors in 0.10700559616088867 seconds
Processed 9 8 predictors in 0.027080059051513672 seconds
Processed 1 9 predictors in 0.004475593566894531 seconds
Total elapsed time : 1.91766357421875 seconds
```

Fungsi ini melakukan pemilihan kombinasi terbaik dari 1 hingga 9 variabel prediktor dalam regresi linier menggunakan RSS. Dari percobaan ini, didapatkan bahwa semakin banyak prediktor, semakin banyak kombinasi yang diuji. Waktu eksekusi meningkat hingga 4 prediktor, lalu mulai menurun. Puncak eksekusi terjadi saat i=4, yaitu 0.5693 detik untuk 126 kombinasi. Setelah i=4, waktu mulai menurun karena semakin sedikit kombinasi tersisa. Total waktu eksekusi untuk seluruh proses adalah 1.917 detik, menunjukkan bahwa percobaan ini berjalan cukup efisien.

```
model_best
✓ 0.0s
```

	RSS	model
1	285.549923	<statsmodels.regression.linear_model.Regressio...
2	215.272521	<statsmodels.regression.linear_model.Regressio...
3	205.45137	<statsmodels.regression.linear_model.Regressio...
4	198.869965	<statsmodels.regression.linear_model.Regressio...
5	191.708199	<statsmodels.regression.linear_model.Regressio...
6	185.236293	<statsmodels.regression.linear_model.Regressio...
7	184.015644	<statsmodels.regression.linear_model.Regressio...
8	184.007534	<statsmodels.regression.linear_model.Regressio...
9	184.007534	<statsmodels.regression.linear_model.Regressio...

Variabel model_best menampilkan model regresi terbaik berdasarkan Residual Sum of Squares (RSS) untuk setiap jumlah prediktor dari 1 hingga 9. Semakin banyak prediktor, nilai RSS semakin kecil. Model dengan 1 prediktor memiliki RSS = 285.55, yang berarti kesalahan prediksi masih tinggi. Model dengan 6 prediktor memiliki RSS = 185.23, menunjukkan peningkatan akurasi yang signifikan. Setelah 7 prediktor, nilai RSS hampir tidak berubah stabil di sekitar 184.01. Penurunan RSS melambat setelah 6 prediktor Dari 6 ke 7 prediktor, RSS turun sedikit 185.23 menjadi 184.02. Dari 7 ke 9 prediktor, RSS hampir tidak berubah. Ini menandakan bahwa menambah prediktor lebih banyak setelah titik tertentu tidak lagi memberikan peningkatan signifikan dalam akurasi.

A. Permodelan Regresi Linier dengan metode Ordinary Least Squares (OLS)

```
print(models_fwd.loc[2]["model"].summary())
✓ 0.0s
```

OLS Regression Results

Dep. Variable:	IPM	R-squared (uncentered):	0.999
Model:	OLS	Adj. R-squared (uncentered):	0.999
Method:	Least Squares	F-statistic:	1.555e+04
Date:	Sun, 16 Mar 2025	Prob (F-statistic):	1.37e-53
Time:	23:40:12	Log-Likelihood:	-86.872
No. Observations:	38	AIC:	177.7
Df Residuals:	36	BIC:	181.0
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Angka Melek Huruf	0.6494	0.033	19.666	0.000	0.582	0.716
APS-SMA	0.1409	0.041	3.428	0.002	0.058	0.224

Omnibus:	3.350	Durbin-Watson:	1.529
Prob(Omnibus):	0.187	Jarque-Bera (JB):	2.982
Skew:	0.675	Prob(JB):	0.225
Kurtosis:	2.756	Cond. No.	15.6

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metode OLS ini untuk memodelkan variabel IPM berdasarkan dua prediktor yakni pada Angka Melek Huruf dan APS-SMA. Hasil regresi menyatakan bahwa nilai R-squared mencapai 0.999 artinya model menjelaskan sebesar 99.9% variabilitas dalam IPM, serta terdapat nilai asj. R-squared sebesar 0.99 setelah penyesuaian terhadap jumlah prediktor model tetap sangat baik. Selanjutnya terdapat nilai P-value pada kedua variabel. Variabel X4 (Angka Melek Huruf) mempunyai nilai sebesar 0.000 artinya Sangat kecil (di bawah 0.05), menunjukkan bahwa variabel ini sangat signifikan dalam mempengaruhi IPM. Lalu pada variabel (X5) APS-SMA menunjukkan nilai P-value Juga lebih kecil dari 0.05, sehingga signifikan dalam model regresi.

```
print(model_best.loc[2,"model"].rsquared)
```
✓ 0.0s

```
0.9988436315655067
```

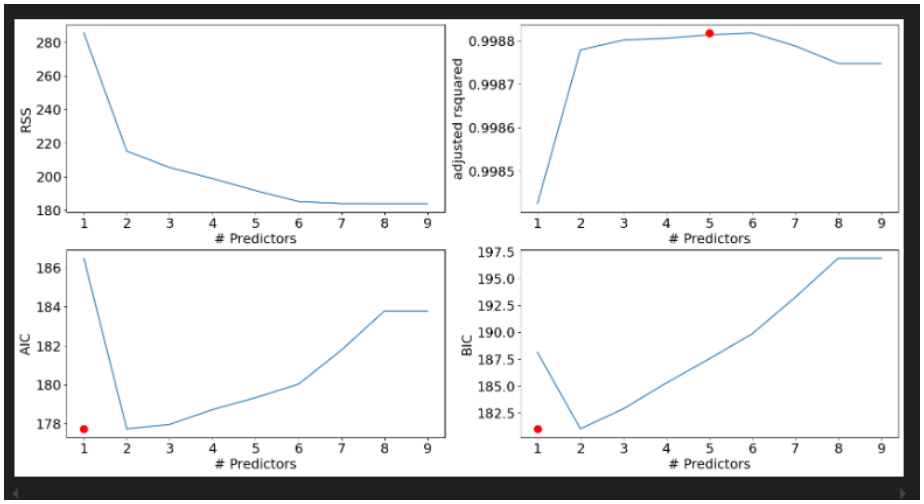
```
model_best.apply(lambda row:row[1].rsquared, axis=1)
```
✓ 0.0s

```
1    0.998466
2    0.998844
3    0.998896
4    0.998932
5    0.998970
6    0.999005
7    0.999012
8    0.999012
9    0.999012
dtype: float64
```

Dari Percobaan ini berarti bahwa hampir 99.84% atau lebih variabilitas dalam data dapat dijelaskan oleh model, yang menunjukkan model sangat baik dalam menjelaskan hubungan antar variabel. Nilai R^2 meningkat dengan bertambahnya jumlah predictor Awalnya R^2 = 0.998466, lalu meningkat hingga sekitar 0.999012. hal Ini menunjukkan bahwa menambahkan lebih banyak prediktor membantu meningkatkan penjelasan variabilitas data. Perubahan R^2 sangat kecil antar model perbedaannya hanya sekitar 0.0005 atau kurang dari satu model ke model lainnya. Hal ini juga menunjukkan bahwa menambahkan prediktor baru mungkin tidak memberikan peningkatan signifikan terhadap performa model. Terdapat Potensi Overfitting R^2 yang sangat tinggi bisa menjadi tanda bahwa model mungkin terlalu menyesuaikan diri dengan data latih, yang bisa menyebabkan performa buruk pada data baru.

B. Menampilkan Plot untuk mengevaluasi performa model regresi

```
plt.figure(figsize=(20,10))
plt.rcParams.update({'font.size':18,'lines.markersize':10})

plt.subplot(2,2,1)

plt.plot(model_best["RSS"])
plt.xlabel("# Predictors")
plt.ylabel("RSS")

rsquared_adj = model_best.apply(lambda row: row[1].rsquared_adj, axis=1)

plt.subplot(2,2,2)
plt.plot(rsquared_adj)
plt.plot(rsquared_adj.argmax(), rsquared_adj.max(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('adjusted rsquared')

aic = model_best.apply(lambda row: row[1].aic, axis=1)

plt.subplot(2,2,3)
plt.plot(aic)
plt.plot(aic.argmin(), aic.min(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('AIC')

bic = model_best.apply(lambda row: row[1].bic, axis=1)

plt.subplot(2,2,4)
plt.plot(bic)
plt.plot(bic.argmin(), bic.min(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('BIC')
```



Dari hasil grafik diatas dapat disimpulkan bahwa:

1. Nlai RSS berdasarkan Jumlah Prediktor menurun dari 1 ke 2 prediktor, lalu terus menurun tetapi dengan laju lebih lambat, hal inimenunjukkan bahwa penampahan prediktor pertama sangat signifikan dalam mengurangi kesalahan. Sehingga Model dengan lebih banyak prediktor lebih baik dalam menyesuaikan data, tetapi pengurangan RSS semakin kecil setelah beberapa prediktor pertama.
2. Interpretasi hasil dari Adjusted R^2 berdasarkan jumlah prediktor yakni Adjusted R^2 meningkat dan mencapai puncak di sekitar 5 prediktor (titik merah), lalu mulai menurun setelahnya. Penurunan ini menunjukkan bahwa penambahan prediktor lebih lanjut tidak memberikan manfaat signifikan dan bisa menyebabkan overfitting.
3. Interpretasi hasil dari AIC berdasarkan jumlah predictor yakni AIC paling rendah pada 1 prediktor (titik merah), lalu meningkat setelahnya. Peningkatan AIC menandakan model menjadi terlalu kompleks tanpa peningkatan signifikan dalam akurasi.
4. Interpretasi Hasil dari BIC berdasarkan Jumlah Prediktor terdapat pada 1 prediktor (titik merah), lalu meningkat secara signifikan setelahnya. Karena BIC lebih ketat dalam menghukum kompleksitas model, kenaikan ini menunjukkan bahwa menambahkan banyak prediktor tidak selau bermanfaat.

C. Pemilihan Metode Subset terbaik

1. Forward Selection

   Forward Selection (Seleksi Maju), Metode ini memulai proses dengan model tanpa variabel prediktor. Selanjutnya, variabel prediktor ditambahkan satu per satu ke dalam model berdasarkan tingkat signifikansinya, biasanya menggunakan nilai p-value. Variabel dengan p-value paling kecil ditambahkan terlebih dahulu. Proses ini berlanjut hingga tidak ada lagi variabel yang memenuhi kriteria signifikansi untuk dimasukkan ke dalam model. Penentuan nilai forward selection dapat menggunakan source code dibawah ini:

INPUT

```
def forward(predictors):
    remaining_predictors = [p for p in X.columns if p not in predictors]

    tic = time.time()

    results = []

    for p in remaining_predictors:
        results.append(processsubset(predictors+[p]))

    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()

    print("Processed", models.shape[0], "predictors in", toc-tic, "seconds")

    return best_model
```

0.0s

```
models_fwd = pd.DataFrame(columns = ["RSS","model"])

tic = time.time()
predictors = []

for i in range(1, len(X.columns)+1):
    models_fwd.loc[i] = forward(predictors)
    predictors = models_fwd.loc[i]["model"].model.exog_names

toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

OUTPUT

```
Processed 9 predictors in 0.05890774726867676 seconds
Processed 8 predictors in 0.036394357681274414 seconds
Processed 7 predictors in 0.04642343521118164 seconds
Processed 6 predictors in 0.029082298278808594 seconds
Processed 5 predictors in 0.025264739990234375 seconds
Processed 4 predictors in 0.021344661712646484 seconds
Processed 3 predictors in 0.023325681686401367 seconds
Processed 2 predictors in 0.014719009399414062 seconds
Processed 1 predictors in 0.005360126495361328 seconds
Total elapsed time : 0.29317378997802734 seconds
```

Dari output yang dihasilkan dapat disimpulkan bahwa semakin banyak prediktor, maka semakin lama waktu pemrosesan datasetnya. Hal ini ditunjukkan pada model dengan 9 prediktor membutuhkan waktu eksekusi sebesar 0.00589 detik, sedangkan model dengan 1 prediktor hanya membutuhkan 0.0054 detik

2. Backward Selection

   Metode ini dimulai dengan model yang mencakup semua variabel prediktor yang tersedia. Kemudian, variabel yang paling tidak signifikan (biasanya dengan p-value terbesar) dihapus dari model. Proses penghapusan ini berlanjut secara iteratif hingga semua variabel yang tersisa dalam model memiliki tingkat signifikansi yang memenuhi kriteria yang ditetapkan

INPUT

```
def backward(predictors):
    tic = time.time()

    results = []

    for combo in itertools.combinations(predictors,len(predictors)-1):
        results.append(processsubset(combo))
    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print('Processed',models.shape[0],'models on',len(predictors)-1, 'predictors in',toc-tic,'seconds')

    return best_model
```

0.0s

```
models_bwd = pd.DataFrame(columns = ["RSS","model"], index = range(1,len(X.columns)))

tic = time.time()
predictors = X.columns

while(len(predictors) > 1):
    models_bwd.loc[len(predictors)-1] = backward(predictors)
    predictors = models_bwd.loc[len(predictors)-1]['model'].model.exog_names
toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

OUTPUT

```
Processed 9 models on 8 predictors in 0.04552435874938965 seconds
Processed 8 models on 7 predictors in 0.028574466705322266 seconds
Processed 7 models on 6 predictors in 0.0265047550201416 seconds
Processed 6 models on 5 predictors in 0.028411388397216797 seconds
Processed 5 models on 4 predictors in 0.023420333862304688 seconds
Processed 4 models on 3 predictors in 0.029214143753051758 seconds
Processed 3 models on 2 predictors in 0.02072763442993164 seconds
Processed 2 models on 1 predictors in 0.017250776290893555 seconds
Total elapsed time : 0.23064589500427246 seconds
```

Dari output yang dihasilkan menunjukkan bahwa Ketika jumlah prediktor menurun maka waktu eksekusi cenderung berkurang, contohnya pada pemrosesan 9 model dengan menggunakan 8 prediktor membutuhkan waktu 0.0455 detik sementara 2 model dengan 1 prediktor hanya membutuhkan 0.0173 detik. Selanjutnya dari hasil pemrosesan itu dapat disimpulkan lagi bahwa waktu pemrosesan tidak bertambah secara linier, terdapat pada pemrosesan 6 model dengan 5 prediktor memiliki waktu 0.0284 detik lebih cepat dibandingkan dengan 3 model pada 3 prediktor sebesar 0.0292 detik

3. Perbandingan Hasil

```
print("FORWARD SELECTION")
print(models_fwd.loc[7,'model'].params)
```
✓ 0.0s

```
FORWARD SELECTION
Angka Melek Huruf       0.612526
APS-SMA                 0.171870
Pengangguran Terbuka    0.142014
PDRB                   -0.147649
Sarana Kesehatan        0.064187
AKB                     0.079231
Persentase Miskin      -0.110171
dtype: float64
```

```
print("BACKWARD SELECTION")
print(models_bwd.loc[7,'model'].params)
```
✓ 0.0s

```
BACKWARD SELECTION
AKB                     0.079231
Sarana Kesehatan        0.064187
Angka Melek Huruf       0.612526
APS-SMA                 0.171870
Persentase Miskin      -0.110171
PDRB                   -0.147649
Pengangguran Terbuka    0.142014
dtype: float64
```

Dari kedua metode pada regresi linier ini, didapatkan Kesimpulan berupa

Forward Selection	Backward Selection
Model dimulai dengan tidak ada rediktor, kemudian model menambahkan variabel satu per satu yang paling meningkatkan performa model	Model dimulai dengan semua prediktor, lalu menghapus satu per satu variabel yang paling tidak signifikan terhadap model

Variabel	Koefisien	Interpretasi Hasil
X4	0.612526	Jika Angka Melek Huruf naik 1 unit, maka variabel target naik 0.612526 unit, dengan asumsi variabel lain tetap.
X5	0.171870	Jika APS-SMA naik 1 unit, maka variabel target naik 0.171870 unit.
X8	0.142014	Jika Pengangguran Terbuka naik 1 unit, maka variabel target naik 0.142014 unit.
X7	-0.147649	Jika PDRB naik 1 unit, maka variabel target turun 0.147649 unit.
X3	0.064187	Jika Sarana Kesehatan naik 1 unit, maka variabel target naik 0.064187 unit.
X1	0.079231	Jika AKB naik 1 unit, maka variabel target naik 0.079231 unit.
X6	-0.110171	Jika Persentase Miskin naik 1 unit, maka variabel target turun 0.110171 unit.

3.1.6 Model Regresi Ridge

A. Menghapus baris naN

INPUT

```
df = pd.read_excel('IPM JATIM 2016.xlsx').dropna().drop('Daerah', axis = 1)
```
✓ 0.0s

```
y = df['IPM']
X = df.drop(['IPM'], axis = 1).astype('float64')
```
✓ 0.0s

```
alphas = 10**np.linspace(10,-2,100)*0.5
alphas
```

OUTPUT

```
array([5.00000000e+09, 3.78231664e+09, 2.86118383e+09, 2.16438064e+09,
       1.63727458e+09, 1.23853818e+09, 9.36908711e+08, 7.08737081e+08,
       5.36133611e+08, 4.05565415e+08, 3.06795364e+08, 2.32079442e+08,
       1.75559587e+08, 1.32804389e+08, 1.00461650e+08, 7.59955541e+07,
       5.74878498e+07, 4.34874501e+07, 3.28966612e+07, 2.48851178e+07,
       1.88246790e+07, 1.42401793e+07, 1.07721735e+07, 8.14875417e+06,
       6.16423370e+06, 4.66301673e+06, 3.52740116e+06, 2.66834962e+06,
       2.01850863e+06, 1.52692775e+06, 1.15506485e+06, 8.73764200e+05,
       6.60970574e+05, 5.00000000e+05, 3.78231664e+05, 2.86118383e+05,
       2.16438064e+05, 1.63727458e+05, 1.23853818e+05, 9.36908711e+04,
       7.08737081e+04, 5.36133611e+04, 4.05565415e+04, 3.06795364e+04,
       2.32079442e+04, 1.75559587e+04, 1.32804389e+04, 1.00461650e+04,
       7.59955541e+03, 5.74878498e+03, 4.34874501e+03, 3.28966612e+03,
       2.48851178e+03, 1.88246790e+03, 1.42401793e+03, 1.07721735e+03,
       8.14875417e+02, 6.16423370e+02, 4.66301673e+02, 3.52740116e+02,
       2.66834962e+02, 2.01850863e+02, 1.52692775e+02, 1.15506485e+02,
       8.73764200e+01, 6.60970574e+01, 5.00000000e+01, 3.78231664e+01,
       2.86118383e+01, 2.16438064e+01, 1.63727458e+01, 1.23853818e+01,
       9.36908711e+00, 7.08737081e+00, 5.36133611e+00, 4.05565415e+00,
       3.06795364e+00, 2.32079442e+00, 1.75559587e+00, 1.32804389e+00,
       1.00461650e+00, 7.59955541e-01, 5.74878498e-01, 4.34874501e-01,
       3.28966612e-01, 2.48851178e-01, 1.88246790e-01, 1.42401793e-01,
       1.07721735e-01, 8.14875417e-02, 6.16423370e-02, 4.66301673e-02,
       3.52740116e-02, 2.66834962e-02, 2.01850863e-02, 1.52692775e-02,
       1.15506485e-02, 8.73764200e-03, 6.60970574e-03, 5.00000000e-03])
```

Interpretasi Hasil

Perintah diatas melakukan pemrosesan dataset IPM JATIM 2016 dengan menghapus baris dengan nilai kosong lalu menghapus kolom “Daerah” karena kolom tersebut tidak dapat digunakan perhitungan secara numerik. Variable dependen (y) diambil dari kolom IPM dan variable independenc(x) merupakan seluruh kolom selain IPM yang nantinya akan dikonversikan kedalam float untuk memastikan tipe data numerik. Disini saya membuat parameter 100 angka yang teristribusi merata antara 10 hingga -2 yang nantinya diubah kedalam skala algoritma dari 10^{10} hingga 10^{-2}

B. Pengujian Dengan Nilai Alpha

INPUT & OUTPUT

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # Normalisasi fitur
```
✓ 0.0s

```
ridge = Ridge()
coefs = []

for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(X_scaled, y)
    coefs.append(ridge.coef_)

np.shape(coefs)
```
✓ 0.2s

```
(100, 9)
```

Interpretasi Hasil

Pemrosesan ini menguji Ridge Regression pada berbagai nilai alpha untuk melihat bagaimana regularisasi mempengaruhi koefisien regresi. Koefisien akan berubah seiring perubahan alpha, menunjukkan efek regularisasi terhadap model. Alpha mengontrol besarnya penalti L2 dalam Ridge Regression. Jika alpha kecil, model mendekati regresi linear biasa. Jika alpha besar, koefisien regresi mengecil, bahkan mendekati nol.

C . Implementasi Nilai Alpha pada Plot

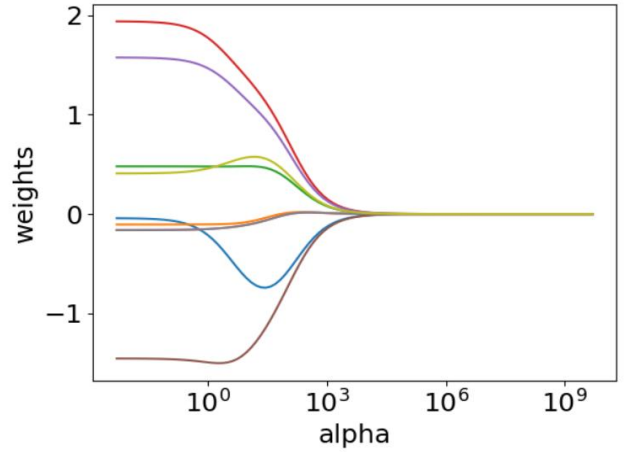
INPUT & OUTPUT Interpretasi Hasil

“Pembuatan Plot”

```
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

✓ 0.6s

```
ext(0, 0.5, 'weights')
```



Plot disamping menunjukkan bagaimana koefisien regresi ridge berubah terhadap nilai alpha dalam skala logaritmik. Sumbu X merupakan nilai alpha dalam Regresi Ridge. Jika nilai alpha kecil maka koefisien semakin besar dan juga jika alpha besar maka semakin besar penalty L2 yang membuat koefisien semakin kecil

D. Partisi dataset training dan testing

```
X_train, X_test , y_train, y_test = train_test_split(X, y,test_size=0.5, random_state=1)

# Normalisasi hanya berdasarkan X_train
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

✓ 0.0s

```
ridge2 = Ridge(alpha=4)  # Hapus normalize=True
ridge2.fit(X_train_scaled, y_train)
pred2 = ridge2.predict(X_test_scaled)

print(pd.Series(ridge2.coef_, index = X.columns))
print(mean_squared_error(y_test, pred2)) # Calculate the test MSE
```

✓ 0.0s

```
AKB                   -0.451789
Keluhan Kesehatan      0.269979
Sarana Kesehatan       0.888106
Angka Melek Huruf      0.772946
APS-SMA                0.506562
Persentase Miskin     -0.202292
PDRB                   0.560133
Pertumbuhan Ekonomi    0.560133
Pengangguran Terbuka   0.657927
dtype: float64
127.21273098157862
```

Melakukan pemisahan dataset training dan dataset testing karena agar terhindar dari overfitting. Overfitting terjadi saat model terlalu kompleks dan menangkap noise dalam data training, sehingga kinerjanya buruk saat diuji pada data baru. Selain itu pemisahan ini juga digunakan untuk mengevaluasi kinerja model secara objektif menggunakan Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared (R^2).

E. Interpretasi Hasil Regresi Ridge dengan alpha = 4

```
ridge2 = Ridge(alpha=4)  # Hapus normalize=True
ridge2.fit(X_train_scaled, y_train)
pred = ridge2.predict(X_test_scaled)
print(pd.Series(ridge2.coef_, index = X.columns))
print(mean_squared_error(y_test, pred2))
```

✓ 0.0s

```
AKB                    -0.451789
Keluhan Kesehatan       0.269979
Sarana Kesehatan        0.888106
Angka Melek Huruf       0.772946
APS-SMA                 0.506562
Persentase Miskin      -0.202292
PDRB                    0.560133
Pertumbuhan Ekonomi     0.560133
Pengangguran Terbuka    0.657927
dtype: float64
127.21273098157862
```

Hasil ini menunjukkan koefisien regresi Ridge dan Mean Squared Error (MSE) dari model dengan alpha = 4. Setiap variabel memiliki nilai koefisien yang menunjukkan pengaruhnya terhadap target (y). Koefisien positif, variabel berkorelasi positif dengan target(peningkatan variabel ini meningkatkan target). Koefisien negatif, variabel berkorelasi negatif dengan target (peningkatan variabel ini menurunkan target).

Sarana Kesehatan (0.888106), berpengaruh paling besar secara positif terhadap target. Angka Melek Huruf (0.772946) , juga memiliki dampak besar dan positif. AKB (-0.451789) memiliki dampak negatif, artinya semakin tinggi AKB, semakin rendah target. Persentase Miskin (-0.202292) juga berdampak negatif. Hasil MSE 127.21 menunjukkan seberapa besar kesalahan prediksi rata-rata model dibandingkan data aktual. Jika MSE rendah model lebih baik tetapi jika MSE tinggi model kurang akurat.

F. Mengecek Alpha Terbaik dengan RidgeCV

```
ridgecv = RidgeCV(alphas=alphas, scoring='neg_mean_squared_error')
ridgecv.fit(X_train_scaled, y_train)

# Cek alpha terbaik
ridgecv.alpha_
```

✓ 0.0s

```
np.float64(21.643806405415308)
```

```
ridge4 = Ridge(alpha=ridgecv.alpha_)
ridge4.fit(X_train_scaled, y_train)

# Hitung MSE pada data test
mse = mean_squared_error(y_test, ridge4.predict(X_test_scaled))
print("Mean Squared Error:", mse)
```

✓ 0.0s

```
Mean Squared Error: 94.64734665694164
```

```
ridge4.fit(X, y)
pd.Series(ridge4.coef_, index = X.columns)
```

✓ 0.0s

```
AKB                    -0.006203
Keluhan Kesehatan      -0.022787
Sarana Kesehatan        0.074316
Angka Melek Huruf       0.341968
APS-SMA                 0.134386
Persentase Miskin      -0.299273
PDRB                   -0.052786
Pertumbuhan Ekonomi    -0.052786
Pengangguran Terbuka    0.198605
```

Alpha terbaik ditemukan sebesar (21.64) ditemukan menggunakan RidgeCV, yang menurunkan MSE dari 127.21 ke 94.65. Regresi Ridge menekan beberapa koefisien lebih kecil (misalnya AKB hampir 0), sehingga model lebih sederhana dan tidak overfitting. Beberapa variabel dengan dampak terbesar:

1. Angka Melek Huruf (Positif besar)
2. Persentase Miskin(Negatif besar)

3.1.7 Model Regresi Lasso

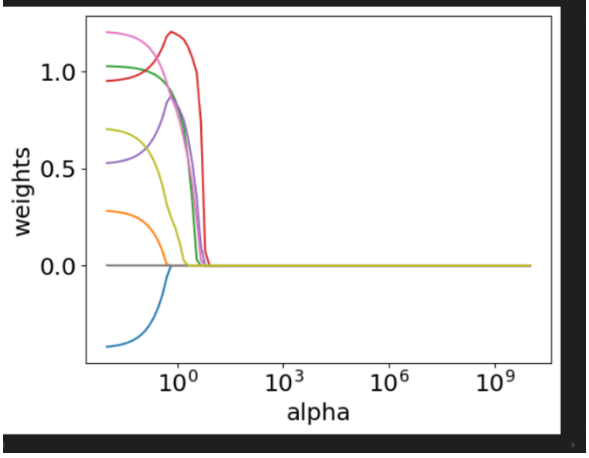
A. Interpretasi Hasil

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
✓ 0.0s

# Lasso Regression tanpa normalize=True
lasso = Lasso(max_iter=10000)

coefs = []
for a in alphas:
    lasso.set_params(alpha=a)
    lasso.fit(X_train_scaled, y_train)
    coefs.append(lasso.coef_)
✓ 0.2s
```

Pemrosesan ini menguji Lasso Regression pada berbagai nilai alpha untuk melihat bagaimana regularisasi mempengaruhi koefisien regresi. Lasso Regression digunakan untuk melakukan seleksi fitur dengan L1 regularization. Hasil alpha dalam Lasso Regression menentukan seberapa banyak fitur yang dipertahankan atau dihapus.

B. Implementasi Nilai Alpha pada Plot

```
ax = plt.gca()
ax.plot(alphas*2, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
✓ 0.4s
ext(0, 0.5, 'weights')
```



Interpretasi Hasil

Plot disamping menunjukkan bagaimana koefisien regresi lasso berubah terhadap nilai alpha dalam skala logaritmik. Sumbu X merupakan nilai alpha dalam Regresi Lasso. Terdapat Alpha optimal dengan nilai di mana sebagian besar fitur yang tidak relevan dihapus, tetapi model masih mempertahankan informasi penting. Nilai alpha terlalu kecil menyebabkan overfitting dengan semua fitur dipertahankan. Serta terdapat Nilai alpha terlalu besar dan mengakibatkan underfitting yang nantinya semua fitur dihapus

C. Mengecek Alpha Terbaik dengan LassoCV

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
✓ 0.0s

lassocv = LassoCV(alphas=None, cv=10, max_iter=100000)
lassocv.fit(X_train_scaled, y_train)

# Gunakan alpha terbaik dari LassoCV
lasso = Lasso(alpha=lassocv.alpha_, max_iter=100000)
lasso.fit(X_train_scaled, y_train)

# Hitung MSE pada data test
mse = mean_squared_error(y_test, lasso.predict(X_test_scaled))
print("Mean Squared Error:", mse)
✓ 0.2s
Mean Squared Error: 62.147680644750764

pd.Series(lasso.coef_, index=X.columns)
✓ 0.0s
AKB                    -0.000000e+00
Keluhan Kesehatan       0.000000e+00
Sarana Kesehatan        7.627253e-01
Angka Melek Huruf       1.176315e+00
APS-SMA                 7.861584e-01
Persentase Miskin      -0.000000e+00
PDRB                    7.188537e-01
Pertumbuhan Ekonomi     2.804774e-16
Pengangguran Terbuka    9.088979e-02
dtype: float64
```

Interpretasi Hasil

Nilai MSE sebesar 62.15 menunjukkan seberapa besar kesalahan prediksi model dibandingkan dengan nilai aktual. Lasso dapat menghapus fitur yang tidak signifikan seperti AKB, Keluhan Kesehatan, Persentase Miskin. Fitur yang tersisa lebih relevan, terutama PDRB dan Pengangguran Terbuka yang memiliki dampak besar. Pertumbuhan Ekonomi memiliki koefisien yang sangat kecil mendekati nol (2.8e-16), sehingga hampir tidak berpengaruh terhadap model.

3.1.8 Mengecek Intercept Model

A. Lasso Regresion

```
lasso_intercept = lasso.intercept_
print("Intercept Lasso:", lasso_intercept)
```

✓ 0.0s

```
Intercept Lasso: 69.13947368421051
```

B. Ridge Regresion

```
ridge_intercept = ridge4.intercept_
print("Intercept Ridge:", ridge_intercept)
```

✓ 0.0s

```
Intercept Ridge: 32.954451659613476
```

Interpretasi Hasil Terbaik

Lasso adalah model yang lebih baik karena memiliki MSE yang jauh lebih rendah sebesar 62.15 dibandingkan Ridge sebesar 94.65. hal Ini menunjukkan bahwa Lasso menghasilkan prediksi yang lebih akurat dalam pemrosesan dataset ini

$$\hat{y} = 69.13947368421051 + 0.7627253X3 + 1.176315X4 + 0.7861584X5 + 0.7188537X7 + 0.00000000000000002804774X8 + 0.09088979X9$$

BAB IV. Percobaan Dataset 2017

4.1 Percobaan Dataset IPM 2017.csv

4.1.1 Install Library

```
import pandas as pd
import numpy as np
import itertools
import matplotlib.pyplot as plt
import time
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import LabelEncoder,scale,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
```

4.1.2 Membaca Dataset

Pada percobaan ini menerapkan model OLS menggunakan data IPM JATIM 2017

```
df = pd.read_excel('IPM Jatim 2017.xlsx')
```

	Daerah	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka	IPM
0	Pacitan	19.91	48.85	1	89.57	75.77	15.42	4.98	4.98	0.85	66.51
1	Ponorogo	22.85	42.08	5	95.02	79.88	11.39	5.10	5.10	3.76	69.26
2	Trenggalek	18.59	45.08	1	95.40	74.40	12.96	5.02	5.02	3.48	68.10
3	Tulungagung	19.18	40.48	11	96.97	80.93	8.04	5.08	5.08	2.27	71.24
4	Blitar	20.84	40.39	6	92.21	65.22	9.80	5.07	5.07	2.99	69.33

Variabel	Simbol
Angka Kematian Bayi	X1
Keluhan Kesehatan	X2
Sarana Kesehatan	X3
Angka Melek Huruf	X4
APS-SMA	X5
Persentase Miskin	X6
PDRB	X7
Pertumbuhan Ekonomi	X8
Pengangguran Terbuka	X9

Kolom Daerah dihapus karena kolom daerah tidak berisikan tipe data numerik sehingga tidak diperlukan dalam perhitungan.

4.1.3 Matriks Korelasi

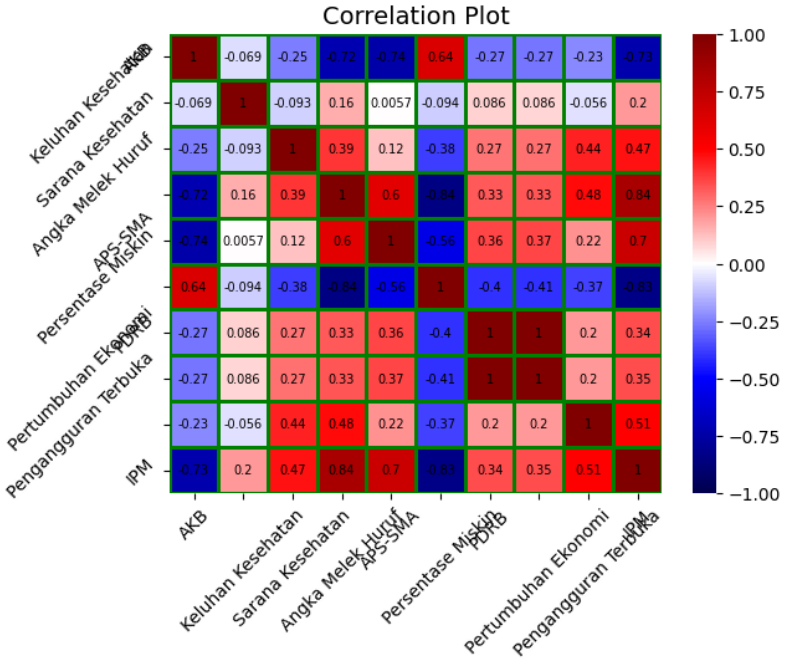
Menentukan matriks korelasi dari semua variabel atau fitur dari dataset yang digunakan

```
kw = df_clean.copy()
```
✓ 0.0s

```
corr_matrix = kw.corr()
corr_matrix
```
✓ 0.0s

	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka	IPM
AKB	1.000000	-0.069423	-0.253502	-0.718054	-0.736358	0.642089	-0.266750	-0.267025	-0.231348	-0.732643
Keluhan Kesehatan	-0.069423	1.000000	-0.092692	0.156759	0.005683	-0.094387	0.086141	0.085580	-0.055550	0.200216
Sarana Kesehatan	-0.253502	-0.092692	1.000000	0.394193	0.122258	-0.382682	0.273267	0.273738	0.436454	0.468447
Angka Melek Huruf	-0.718054	0.156759	0.394193	1.000000	0.602393	-0.844428	0.327688	0.328616	0.481280	0.839223
APS-SMA	-0.736358	0.005683	0.122258	0.602393	1.000000	-0.562592	0.364971	0.365528	0.217915	0.703475
Persentase Miskin	0.642089	-0.094387	-0.382682	-0.844428	-0.562592	1.000000	-0.403823	-0.405258	-0.371376	-0.833164
PDRB	-0.266750	0.086141	0.273267	0.327688	0.364971	-0.403823	1.000000	0.999989	0.202090	0.344938
Pertumbuhan Ekonomi	-0.267025	0.085580	0.273738	0.328616	0.365528	-0.405258	0.999989	1.000000	0.202116	0.345676
Pengangguran Terbuka	-0.231348	-0.055550	0.436454	0.481280	0.217915	-0.371376	0.202090	0.202116	1.000000	0.507358
IPM	-0.732643	0.200216	0.468447	0.839223	0.703475	-0.833164	0.344938	0.345676	0.507358	1.000000

```
sns.heatmap(corr_matrix,
            cmap = 'seismic',
            linewidth = 0.75,
            linecolor = 'green',
            cbar = True,
            vmin = -1,
            vmax = 1,
            annot = True,
            annot_kws = {'size': 7,'color':'black'})
plt.tick_params(labelsize = 10, rotation = 45)
plt.title('Correlation Plot',size = 14)
```



Dari output yang dihasilkan dapat disimpulkan bahwa nilai korelasi yang cenderung warna merah berarti cukup tinggi(korelasi positif). Warna biru juga mempunyai korelasi yang cukup tinggi (korelasi negatif). Nilai korelasi diatas 0,5 merupakan nilai korelasi yang paling bagus. Akan tetapi apabila mendekati 1 harus dicek ulang hubungan antara variabel independent karena takutnya apabila sudah mendekati 1 akan terjasi multikolinearitas.

4.1.4 Inisiasi Variabel Responden & Prediktor

```
y = df_clean['IPM']  # Target variable
X = df_clean.drop(['IPM'], axis=1)  # Feature matrix
X.head()
```

✓ 0.0s

	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka
0	19.91	48.85	1	89.57	75.77	15.42	4.98	4.98	0.85
1	22.85	42.08	5	95.02	79.88	11.39	5.10	5.10	3.76
2	18.59	45.08	1	95.40	74.40	12.96	5.02	5.02	3.48
3	19.18	40.48	11	96.97	80.93	8.04	5.08	5.08	2.27
4	20.84	40.39	6	92.21	65.22	9.80	5.07	5.07	2.99

Dari percobaan diatas didapatkan bahwa Variabel Responden (Y) adalah kolom IPM dan Variabel Prediktor (X) adalah kolom AKB, Keluhan Kesehatan, Angka Melek Huruf, APS-SMA, Persentase Miskin, PDRB, Pertumbuhan Ekonomi, Pengangguran Terbuka. Dari dataset ini dapat dianalisis seberapa pengaruhnya variabel prediktor(X) terhadap variabel responden(Y).

4.1.5 Regresi Linier

```
def processsubset(featureset):
    model = sm.OLS(y,X[list(featureset)])
    regr = model.fit()
    RSS = ((regr.predict(X[list(featureset)])-y)**2).sum()
    return {"model":regr,"RSS":RSS}
```

✓ 0.0s

Fungsi ini berguna untuk mengevaluasi kombinasi variabel prediktor dalam regresi linear. Nilai RSS yang lebih kecil menunjukkan model yang lebih baik dalam memprediksi data. Hasil dari variabel model dapat digunakan untuk analisis lebih lanjut seperti uji signifikansi, R-squared, dan interpretasi koefisien regresi.

```
def getbest(k):
    tic = time.time()
    results = []
    for combo in itertools.combinations(X.columns,k):
        results.append(processsubset(combo))
    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print("Processed", models.shape[0], k, "predictors in", toc-tic, "seconds")

    return best_model
```

Fungsi getbest(k) berguna untuk mencari kombinasi fitur terbaik dari ukuran k berdasarkan RSS terkecil. Digunakan dalam metode seleksi fitur untuk menentukan kombinasi optimal dalam regresi linear. Hasilnya bisa dianalisis lebih lanjut untuk interpretasi koefisien dan signifikansi statistik.

```
model_best = pd.DataFrame(columns = ["RSS","model"])

tic = time.time()
for i in range(1,10):
    model_best.loc[i] = getbest(i)
toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

✓ 1.2s

```
Processed 9 1 predictors in 0.020838260650634766 seconds
Processed 36 2 predictors in 0.0476686954498291 seconds
Processed 84 3 predictors in 0.095016241073608 4 seconds
Processed 126 4 predictors in 0.277667760848999 seconds
Processed 126 5 predictors in 0.3372511863708496 seconds
Processed 84 6 predictors in 0.2323300838470459 seconds
Processed 36 7 predictors in 0.1381368637084961 seconds
Processed 9 8 predictors in 0.032073020935058594 seconds
Processed 1 9 predictors in 0.005754709243774414 seconds
Total elapsed time : 1.210237741470337 seconds
```

Fungsi ini melakukan pemilihan kombinasi terbaik dari 1 hingga 9 variabel prediktor dalam regresi linier menggunakan RSS. Dari percobaan ini, didapatkan bahwa semakin banyak prediktor, semakin banyak kombinasi

```
model_best
✓ 0.0s
```

	RSS	model
1	330.54743	<statsmodels.regression.linear_model.Regressio...
2	249.740821	<statsmodels.regression.linear_model.Regressio...
3	206.548128	<statsmodels.regression.linear_model.Regressio...
4	186.863428	<statsmodels.regression.linear_model.Regressio...
5	175.028559	<statsmodels.regression.linear_model.Regressio...
6	166.604749	<statsmodels.regression.linear_model.Regressio...
7	164.722151	<statsmodels.regression.linear_model.Regressio...
8	162.735867	<statsmodels.regression.linear_model.Regressio...
9	162.363416	<statsmodels.regression.linear_model.Regressio...

Variabel model_best menampilkan model regresi terbaik berdasarkan Residual Sum of Squares (RSS) untuk setiap jumlah prediktor dari 1 hingga 9. Semakin banyak prediktor, nilai RSS semakin kecil. Model dengan 1 prediktor memiliki RSS 330.54743, yang berarti kesalahan prediksi masih tinggi. Model dengan 6 prediktor memiliki RSS = 166.604749, menunjukkan peningkatan akurasi yang signifikan. Setelah 7 prediktor, nilai RSS hampir tidak berubah (stabil di sekitar 162). Penurunan RSS melambat setelah 6 prediktor Dari 6 ke 7 prediktor, RSS turun sedikit. Dari 7 ke 9 prediktor, RSS hampir tidak berubah. Ini menandakan bahwa menambah prediktor lebih banyak setelah titik tertentu tidak lagi memberikan peningkatan signifikan dalam akurasi.

A. Permodelan Regresi Linier dengan metode Ordinary Least Squares (OLS)

```
print(model_best.loc[2]["model"].summary())
✓ 0.0s
```

OLS Regression Results

Dep. Variable:	IPM	R-squared (uncentered):	0.999
Model:	OLS	Adj. R-squared (uncentered):	0.999
Method:	Least Squares	F-statistic:	1.361e+04
Date:	Mon, 17 Mar 2025	Prob (F-statistic):	1.49e-52
Time:	12:53:07	Log-Likelihood:	-89.694
No. Observations:	38	AIC:	183.4
Df Residuals:	36	BIC:	186.7
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Angka Melek Huruf	0.6255	0.040	15.472	0.000	0.544	0.708
APS-SMA	0.1711	0.050	3.413	0.002	0.069	0.273

Omnibus:	5.543	Durbin-Watson:	1.247
Prob(Omnibus):	0.063	Jarque-Bera (JB):	4.118
Skew:	0.717	Prob(JB):	0.128
Kurtosis:	3.737	Cond. No.	17.8

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metode OLS ini untuk memodelkan variabel IPM berdasarkan dua prediktor yakni pada Angka Melek Huruf dan APS-SMA. Hasil regresi menyatakan bahwa nilai R-squared mencapai 0.999 artinya model menjelaskan sebesar 99.9% variabilitas dalam IPM, serta terdapat nilai asj. R-squared sebesar 0.99 setelah penyesuaian terhadap jumlah prediktor model tetap sangat baik. Selanjutnya terdapat nilai P-value pada kedua variabel. Variabel X4 (Angka Melek Huruf) mempunyai nilai sebesar 0.000 artinya Sangat kecil (di bawah 0.05), menunjukkan bahwa variabel ini sangat signifikan dalam mempengaruhi IPM. Lalu pada variabel (X5) APS-SMA menunjukkan nilai P-value Juga lebih kecil dari 0.05, sehingga signifikan dalam model regresi.

```
print(model_best.loc[2,"model"].rsquared)
```
✓ 0.0s

```
0.9986794748400373
```

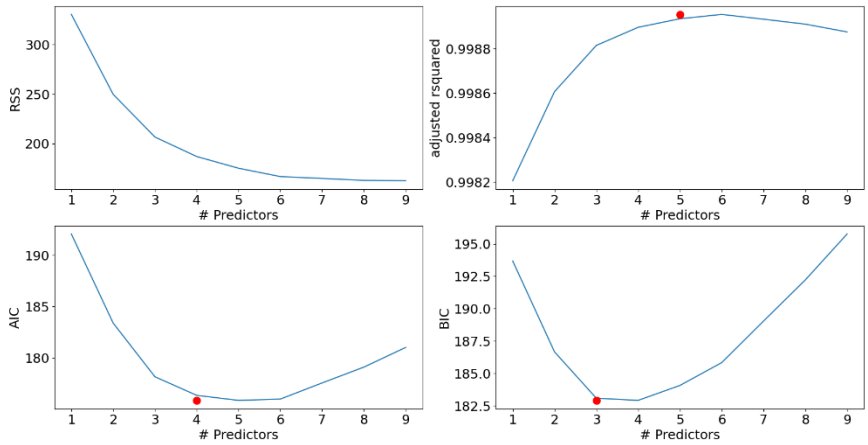
```
model_best.apply(lambda row:row[1].rsquared, axis=1)
```
✓ 0.0s

```
1    0.998252
2    0.998679
3    0.998908
4    0.999012
5    0.999075
6    0.999119
7    0.999129
8    0.999140
9    0.999141
dtype: float64
```

Dari Percobaan ini berarti bahwa hampir 99.86% atau lebih variabilitas dalam data dapat dijelaskan oleh model, yang menunjukkan model sangat baik dalam menjelaskan hubungan antar variabel. Nilai R^2 meningkat dengan bertambahnya jumlah predictor Awalnya R^2 = 0.998679, lalu meningkat hingga sekitar 0.999141. hal Ini menunjukkan bahwa menambahkan lebih banyak prediktor membantu meningkatkan penjelasan variabilitas data. Perubahan R^2 sangat kecil antar model perbedaannya hanya sekitar 0.0005 atau kurang dari satu model ke model lainnya. Hal ini juga menunjukkan bahwa menambahkan prediktor baru mungkin tidak memberikan peningkatan signifikan terhadap performa model. Terdapat Potensi Overfitting R^2 yang sangat tinggi bisa menjadi tanda bahwa model mungkin terlalu menyesuaikan diri dengan data latih, yang bisa menyebabkan performa buruk pada data baru.

B. Menampilkan Plot untuk mengevaluasi performa model regresi

```
plt.figure(figsize=(20,10))
plt.rcParams.update({'font.size':18,'lines.markersize':10})

plt.subplot(2,2,1)

plt.plot(model_best["RSS"])
plt.xlabel("# Predictors")
plt.ylabel("RSS")

rsquared_adj = model_best.apply(lambda row: row[1].rsquared_adj, axis=1)

plt.subplot(2,2,2)
plt.plot(rsquared_adj)
plt.plot(rsquared_adj.argmax(), rsquared_adj.max(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('adjusted rsquared')

aic = model_best.apply(lambda row: row[1].aic, axis=1)

plt.subplot(2,2,3)
plt.plot(aic)
plt.plot(aic.argmin(), aic.min(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('AIC')

bic = model_best.apply(lambda row: row[1].bic, axis=1)

plt.subplot(2,2,4)
plt.plot(bic)
plt.plot(bic.argmin(), bic.min(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('BIC')
```



1. Residual Sum of Squares (RSS) (Kiri Atas)
- RSS menurun seiring bertambahnya jumlah prediktor.
- Ini menunjukkan bahwa model semakin baik dalam menyesuaikan data saat lebih banyak prediktor ditambahkan.
- Namun, setelah sekitar 5 prediktor, penurunan RSS tidak lagi signifikan.
2. Adjusted R-squared (Kanan Atas)
- Nilai adjusted R^2 meningkat seiring bertambahnya prediktor dan mencapai titik maksimum sekitar 5 prediktor (ditandai titik merah).
- Setelah itu, nilainya mulai stagnan atau menurun sedikit, menunjukkan bahwa menambah lebih banyak variabel tidak meningkatkan model secara signifikan.
3. Akaike Information Criterion (AIC) (Kiri Bawah)
- AIC mencapai nilai minimum sekitar 4 prediktor (ditandai titik merah), lalu meningkat kembali.
- Ini menunjukkan bahwa model dengan 4 prediktor adalah yang optimal berdasarkan keseimbangan antara akurasi dan kompleksitas model.
4. Bayesian Information Criterion (BIC) (Kanan Bawah)
- BIC juga mencapai nilai minimum pada 3 atau 4 prediktor sebelum meningkat kembali.
- Karena BIC lebih ketat dalam penalti terhadap kompleksitas model dibandingkan AIC, ini mengindikasikan bahwa model dengan 3 atau 4 prediktor adalah pilihan terbaik untuk menghindari overfitting.

C. Pemilihan Metode Subset terbaik

1. Forward Selection

Forward Selection (Seleksi Maju), Metode ini memulai proses dengan model tanpa variabel prediktor. Selanjutnya, variabel prediktor ditambahkan satu per satu ke dalam model berdasarkan tingkat signifikansinya, biasanya menggunakan nilai p-value. Variabel dengan p-value paling kecil ditambahkan terlebih dahulu. Proses ini berlanjut hingga tidak ada lagi variabel yang memenuhi kriteria signifikansi untuk dimasukkan ke dalam model. Penentuan nilai forward selection dapat menggunakan source code dibawah ini:

INPUT

```
def forward(predictors):
    remaining_predictors = [p for p in X.columns if p not in predictors]

    tic = time.time()

    results = []

    for p in remaining_predictors:
        results.append(processsubset(predictors+[p]))

    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()

    print("Processed", models.shape[0], "predictors in", toc-tic, "seconds")

    return best_model
```

0.0s

```
models_fwd = pd.DataFrame(columns = ["RSS","model"])

tic = time.time()
predictors = []

for i in range(1, len(X.columns)+1):
    models_fwd.loc[i] = forward(predictors)
    predictors = models_fwd.loc[i]["model"].model.exog_names

toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

OUTPUT

0.2s

```
Processed 9 predictors in 0.05142807960510254 seconds
Processed 8 predictors in 0.03957748413085937 seconds
Processed 7 predictors in 0.020026445388793945 seconds
Processed 6 predictors in 0.020514249801635742 seconds
Processed 5 predictors in 0.019639968872070312 seconds
Processed 4 predictors in 0.014578819274902344 seconds
Processed 3 predictors in 0.013997793197631836 seconds
Processed 2 predictors in 0.006705284118652344 seconds
Processed 1 predictors in 0.004578113555908203 seconds
Total elapsed time : 0.2113852500915273 seconds
```

Dari output yang dihasilkan dapat disimpulkan bahwa semakin banyak prediktor, maka semakin lama waktu pemrosesan datasetnya. Hal ini ditunjukkan pada model dengan 9 prediktor membutuhkan waktu eksekusi sebesar 0.00514280 detik, sedangkan model dengan 1 prediktor hanya membutuhkan 0.004578 detik

2. Backward Selection

Metode ini dimulai dengan model yang mencakup semua variabel prediktor yang tersedia. Kemudian, variabel yang paling tidak signifikan (biasanya dengan p-value terbesar) dihapus dari model. Proses penghapusan ini berlanjut secara iteratif hingga semua variabel yang tersisa dalam model memiliki tingkat signifikansi yang memenuhi kriteria yang ditetapkan

INPUT

```
def backward(predictors):
    tic = time.time()

    results = []

    for combo in itertools.combinations(predictors,len(predictors)-1):
        results.append(processsubset(combo))
    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print('Processed',models.shape[0],'models on',len(predictors)-1, 'predictors in',toc-tic,'seconds')

    return best_model
```

0.0s

```
models_bwd = pd.DataFrame(columns = ["RSS","model"], index = range(1,len(X.columns)))

tic = time.time()
predictors = X.columns

while(len(predictors) > 1):
    models_bwd.loc[len(predictors)-1] = backward(predictors)
    predictors = models_bwd.loc[len(predictors)-1]['model'].model.exog_names
toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

OUTPUT

```
Processed 9 models on 8 predictors in 0.06732821464538574 seconds
Processed 8 models on 7 predictors in 0.03290843963623047 seconds
Processed 7 models on 6 predictors in 0.02506422996520996 seconds
Processed 6 models on 5 predictors in 0.02295541763305664 seconds
Processed 5 models on 4 predictors in 0.020171403884887695 seconds
Processed 4 models on 3 predictors in 0.014421939849853516 seconds
Processed 3 models on 2 predictors in 0.009612798690795898 seconds
Processed 2 models on 1 predictors in 0.006513357162475586 seconds
Total elapsed time : 0.20711088180541992 seconds
```

Dari output yang dihasilkan menunjukkan bahwa Ketika jumlah prediktor menurun maka waktu eksekusi cenderung berkurang, contohnya pada pemrosesan 9 model dengan menggunakan 8 prediktor membutuhkan waktu 0.0455 detik sementara 2 model dengan 1 prediktor hanya membutuhkan 0.0173 detik. Selanjutnya dari hasil pemrosesan itu dapat disimpulkan lagi bahwa waktu pemrosesan tidak bertambah secara linier, terdapat pada pemrosesan 6 model dengan 5 prediktor memiliki waktu 0.0284 detik lebih cepat dibandingkan dengan 3 model pada 3 prediktor sebesar 0.0292 detik

3. Perbandingan Hasil

```
print("FORWARD SELECTION")
print(models_fwd.loc[7,'model'].params)
```

0.0s

```
FORWARD SELECTION
Angka Melek Huruf       0.550377
APS-SMA                 0.179408
Sarana Kesehatan        0.123981
Keluhan Kesehatan       0.136434
Pengangguran Terbuka    0.443135
Persentase Miskin      -0.107761
Pertumbuhan Ekonomi    -0.242013
dtype: float64
```

```
print("BACKWARD SELECTION")
print(models_bwd.loc[7,'model'].params)
```

0.0s

```
BACKWARD SELECTION
Keluhan Kesehatan       0.136434
Sarana Kesehatan        0.123981
Angka Melek Huruf       0.550377
APS-SMA                 0.179408
Persentase Miskin      -0.107761
Pertumbuhan Ekonomi    -0.242013
Pengangguran Terbuka    0.443135
dtype: float64
```

Dari kedua metode pada regresi linier ini, didapatkan Kesimpulan berupa

Forward Selection	Backward Selection
Model dimulai dengan tidak ada rediktor, kemudian model menambahkan variabel satu per satu yang paling meningkatkan performa model	Model dimulai dengan semua prediktor, lalu menghapus satu per satu variabel yang paling tidak signifikan terhadap model

4.1.6 Model Regresi Ridge

A. Menghapus baris naN

INPUT

```
df = pd.read_excel('IPM JATIM 2017.xlsx').dropna().drop('Daerah', axis = 1)
✓ 0.0s

y = df['IPM']
X = df.drop(['IPM'], axis = 1).astype('float64')
✓ 0.0s

alphas = 10**np.linspace(10,-2,100)*0.5
alphas
✓ 0.0s
```

OUTPUT

```
array([5.00000000e+09, 3.78231664e+09, 2.86118383e+09, 2.16438064e+09,
       1.63727458e+09, 1.23853818e+09, 9.36908711e+08, 7.08737081e+08,
       5.36133611e+08, 4.05565415e+08, 3.06795364e+08, 2.32079442e+08,
       1.75559587e+08, 1.32804389e+08, 1.00461650e+08, 7.59955541e+07,
       5.74878498e+07, 4.34874501e+07, 3.28966612e+07, 2.48851178e+07,
       1.88246790e+07, 1.42401793e+07, 1.07721735e+07, 8.14875417e+06,
       6.16423370e+06, 4.66301673e+06, 3.52740116e+06, 2.66834962e+06,
       2.01850863e+06, 1.52692775e+06, 1.15506485e+06, 8.73764200e+05,
       6.60970574e+05, 5.00000000e+05, 3.78231664e+05, 2.86118383e+05,
       2.16438064e+05, 1.63727458e+05, 1.23853818e+05, 9.36908711e+04,
       7.08737081e+04, 5.36133611e+04, 4.05565415e+04, 3.06795364e+04,
       2.32079442e+04, 1.75559587e+04, 1.32804389e+04, 1.00461650e+04,
       7.59955541e+03, 5.74878498e+03, 4.34874501e+03, 3.28966612e+03,
       2.48851178e+03, 1.88246790e+03, 1.42401793e+03, 1.07721735e+03,
       8.14875417e+02, 6.16423370e+02, 4.66301673e+02, 3.52740116e+02,
       2.66834962e+02, 2.01850863e+02, 1.52692775e+02, 1.15506485e+02,
       8.73764200e+01, 6.60970574e+01, 5.00000000e+01, 3.78231664e+01,
       2.86118383e+01, 2.16438064e+01, 1.63727458e+01, 1.23853818e+01,
       9.36908711e+00, 7.08737081e+00, 5.36133611e+00, 4.05565415e+00,
       3.06795364e+00, 2.32079442e+00, 1.75559587e+00, 1.32804389e+00,
       1.00461650e+00, 7.59955541e-01, 5.74878498e-01, 4.34874501e-01,
       3.28966612e-01, 2.48851178e-01, 1.88246790e-01, 1.42401793e-01,
       1.07721735e-01, 8.14875417e-02, 6.16423370e-02, 4.66301673e-02,
       3.52740116e-02, 2.66834962e-02, 2.01850863e-02, 1.52692775e-02,
       1.15506485e-02, 8.73764200e-03, 6.60970574e-03, 5.00000000e-03])
```

Interpretasi Hasil

Perintah diatas melakukan pemrosesan dataset IPM JATIM 2016 dengan menghapus baris dengan nilai kosong lalu menghapus kolom “Daerah” karena kolom tersebut tidak dapat digunakan perhitungan secara numerik. Variable dependen (y) diambil dari kolom IPM dan variable independenc(x) merupakan seluruh kolom selain IPM yang nantinya akan dikonversikan kedalam float untuk memastikan tipe data numerik. Disini saya membuat parameter 100 angka yang teristribusi merata antara 10 hingga -2 yang nantinya diubah kedalam skala algoritma dari 10^{10} hingga 10^{-2}

B. Pengujian Dengan Nilai Alpha

INPUT & OUTPUT

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # Normalisasi fitur
✓ 0.0s

ridge = Ridge()
coefs = []

for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(X_scaled, y)
    coefs.append(ridge.coef_)

np.shape(coefs)
✓ 0.2s

(100, 9)
```

Interpretasi Hasil

Pemrosesan ini menguji Ridge Regression pada berbagai nilai alpha untuk melihat bagaimana regularisasi mempengaruhi koefisien regresi. Koefisien akan berubah seiring perubahan alpha, menunjukkan efek regularisasi terhadap model. Alpha mengontrol besarnya penalti L2 dalam Ridge Regression. Jika alpha kecil, model mendekati regresi linear biasa. Jika alpha besar, koefisien regresi mengecil, bahkan mendekati nol.

C . Implementasi Nilai Alpha pada Plot

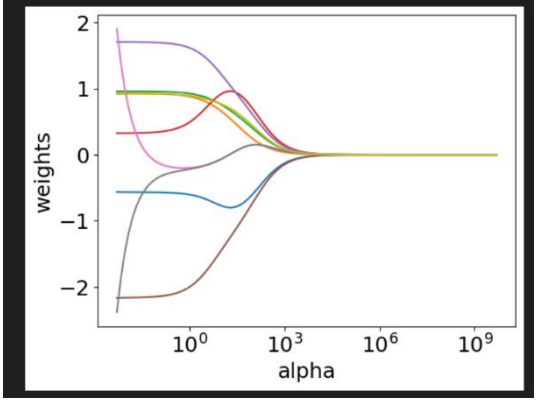
INPUT & OUTPUT

“Pembuatan Plot”

```
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

✓ 0.6s

```
Text(0, 0.5, 'weights')
```



Interpretasi Hasil

Plot disamping menunjukkan bagaimana koefisien regresi ridge berubah terhadap nilai alpha dalam skala logaritmik. Sumbu X merupakan nilai alpha dalam Regresi Ridge. Jika nilai alpha kecil maka koefisien semakin besar dan juga jika alpha besar maka semakin besar penalty L2 yang membuat koefisien semakin kecil

D. Partisi dataset training dan testing

```
X_train, X_test , y_train, y_test = train_test_split(X, y,test_size=0.5, random_state=1)

# Normalisasi hanya berdasarkan X_train
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[33] ✓ 0.0s

Melakukan pemisahan dataset training dan dataset testing karena agar terhindar dari overfitting. Overfitting terjadi saat model terlalu kompleks dan menangkap noise dalam data training, sehingga kinerjanya buruk saat diuji pada data baru. Selain itu pemisahan ini juga digunakan untuk mengevaluasi kinerja model secara objektif menggunakan Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared (R^2).

E. Interpretasi Hasil Regresi Ridge dengan alpha = 4

```
ridge2 = Ridge(alpha=4)
ridge2.fit(X_train_scaled, y_train)
pred2 = ridge2.predict(X_test_scaled)

print(pd.Series(ridge2.coef_, index = X.columns))
print(mean_squared_error(y_test, pred2)) # Calculate the test MSE
```

✓ 0.0s

```
AKB                      -0.408892
Keluhan Kesehatan         0.638207
Sarana Kesehatan          1.277483
Angka Melek Huruf         0.886642
APS-SMA                   0.920561
Persentase Miskin        -0.406872
PDRB                      0.373896
Pertumbuhan Ekonomi       0.376209
Pengangguran Terbuka      0.253512
dtype: float64
13.994737728463743
```

Hasil ini menunjukkan koefisien regresi Ridge dan Mean Squared Error (MSE) dari model dengan alpha = 4. Setiap variabel memiliki nilai koefisien yang menunjukkan pengaruhnya terhadap target (y). Koefisien positif, variabel berkorelasi positif dengan target(peningkatan variabel ini meningkatkan target). Koefisien negatif, variabel berkorelasi negatif dengan target (peningkatan variabel ini menurunkan target).

F. Mengecek Alpha Terbaik dengan RidgeCV

```
ridgecv = RidgeCV(alphas=alphas, scoring='neg_mean_squared_error')
ridgecv.fit(X_train_scaled, y_train)

# Cek alpha terbaik
ridgecv.alpha_
```
✓ 0.1s

```
np.float64(16.372745814388658)
```

+ Code + Markdown

```
ridge4 = Ridge(alpha=ridgecv.alpha_)  # Hapus normalize=True
ridge4.fit(X_train_scaled, y_train)

# Hitung MSE pada data test
mse = mean_squared_error(y_test, ridge4.predict(X_test_scaled))
print("Mean Squared Error:", mse)
```
✓ 0.0s

```
Mean Squared Error: 17.468117081215546
```

```
ridge4.fit(X, y)
pd.Series(ridge4.coef_, index = X.columns)
```
✓ 0.0s

AKB	-0.048566
Keluhan Kesehatan	0.134642
Sarana Kesehatan	0.134976
Angka Melek Huruf	0.123686
APS-SMA	0.163130
Persentase Miskin	-0.422114
PDRB	-0.169248
Pertumbuhan Ekonomi	-0.170440
Pengangguran Terbuka	0.513711

Alpha terbaik ditemukan sebesar 16.3727 ditemukan menggunakan RidgeCV. Variabel yang memiliki dampak besar yakni pengangguran terbuka sebesar 0.513711 dengan pengaruh paling positif dan Pertumbuhan ekonomi sebesar -0.172114 berpengaruh negatif

4.1.7 Model Regresi Lasso

A. Interpretasi Hasil

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```
✓ 0.0s

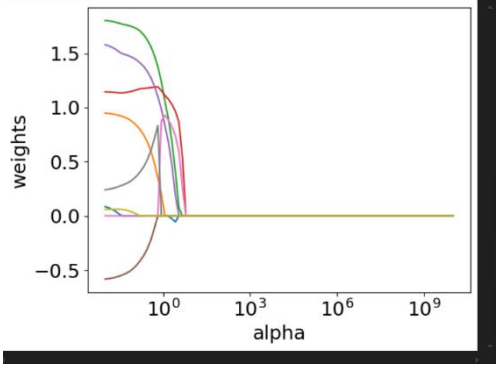
```
# Lasso Regression tanpa normalize=True
lasso = Lasso(max_iter=10000)

coefs = []
for a in alphas:
    lasso.set_params(alpha=a)
    lasso.fit(X_train_scaled, y_train)
    coefs.append(lasso.coef_)
```
✓ 0.2s

Pemrosesan ini menguji Lasso Regression pada berbagai nilai alpha untuk melihat bagaimana regularisasi mempengaruhi koefisien regresi. Lasso Regression digunakan untuk melakukan seleksi fitur dengan L1 regularization. Hasil alpha dalam Lasso Regression menentukan seberapa banyak fitur yang dipertahankan atau dihapus.

B. Implementasi Nilai Alpha pada Plot

```
ax = plt.gca()
ax.plot(alphas*2, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```
✓ 0.3s Python

```
ext(0, 0.5, 'weights')
```



Interpretasi Hasil

Plot disamping menunjukkan bagaimana koefisien regresi lasso berubah terhadap nilai alpha dalam skala logaritmik. Sumbu X merupakan nilai alpha dalam Regresi Lasso. Terdapat Alpha optimal dengan nilai di mana sebagian besar fitur yang tidak relevan dihapus, tetapi model masih mempertahankan informasi penting. Nilai alpha terlalu kecil menyebabkan overfitting dengan semua fitur dipertahankan. Serta terdapat Nilai alpha terlalu besar dan mengakibatkan underfitting yang nantinya semua fitur dihapus

C. Mengecek Alpha Terbaik dengan LassoCV

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
✓ 0.0s
                                    + Code  + Markdown

lassocv = LassoCV(alphas=None, cv=10, max_iter=100000)
lassocv.fit(X_train_scaled, y_train)

# Gunakan alpha terbaik dari LassoCV
lasso = Lasso(alpha=lassocv.alpha_, max_iter=100000)
lasso.fit(X_train_scaled, y_train)

# Hitung MSE pada data test
mse = mean_squared_error(y_test, lasso.predict(X_test_scaled))
print("Mean Squared Error:", mse)
✓ 0.2s
Mean Squared Error: 17.346255606433


pd.Series(lasso.coef_, index=X.columns)
✓ 0.0s
AKB                      -0.000000
Keluhan Kesehatan         0.324440
Sarana Kesehatan          1.352677
Angka Melek Huruf         1.184651
APS-SMA                   1.081040
Persentase Miskin        -0.000000
PDRB                      0.000000
Pertumbuhan Ekonomi       0.839722
Pengangguran Terbuka      0.000000
dtype: float64
```

Interpretasi Hasil

Nilai MSE sebesar 17.346 menunjukkan seberapa besar kesalahan prediksi model dibandingkan dengan nilai aktual. Lasso dapat menghapus fitur yang tidak signifikan seperti AKB, Persentase Miskin, PDRB, Pengangguran Terbuka. Fitur yang tersisa lebih relevan terhadap model.

4.1.8 Mengecek Intercept Model

A. Lasso Regresion

```
lasso_intercept = lasso.intercept_
print("Intercept Lasso:", lasso_intercept)
✓ 0.0s
Intercept Lasso: 69.7836842105263
```

B. Ridge Regresion

```
ridge_intercept = ridge4.intercept_
print("Intercept Ridge:", ridge_intercept)
✓ 0.0s
Intercept Ridge: 45.95091301030947
```

Interpretasi Hail Terbaik

Ridge Regression adalah model yang lebih baik karena memiliki MSE yang jauh lebih rendah sebesar 45.95091 dibandingkan Lasso sebesar 69.783. hal Ini menunjukkan bahwa Ridge menghasilkan prediksi yang lebih akurat dalam pemrosesan dataset ini

BAB V. Percobaan Dataset 2018

5.1 Percobaan Dataset IPM 2018.csv

5.1.1 Install Library

```
import pandas as pd
import numpy as np
import itertools
import matplotlib.pyplot as plt
import time
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import LabelEncoder,scale,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
```

5.1.2 Membaca Dataset

Pada percobaan ini menerapkan model OLS menggunakan data IPM JATIM 2018

```
df = pd.read_excel('IPM Jatim 2018.xlsx')
df.head()
```

✓ 0.4s

	Daerah	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka	IPM
0	Pacitan	19.71	42.07	3	92.57	76.12	14.19	5.51	5.47	1.39	67.33
1	Ponorogo	22.21	53.78	5	89.11	77.50	10.36	5.31	5.27	3.77	69.91
2	Trenggalek	18.28	48.01	1	94.41	76.78	12.02	5.03	5.03	4.12	68.71
3	Tulungagung	18.91	48.32	12	96.84	78.54	7.27	5.21	5.21	2.53	71.99
4	Blitar	20.67	41.78	7	94.49	68.80	9.72	5.10	5.10	3.38	69.93

Variabel	Simbol
Angka Kematian Bayi	X1
Keluhan Kesehatan	X2
Sarana Kesehatan	X3
Angka Melek Huruf	X4
APS-SMA	X5
Persentase Miskin	X6
PDRB	X7
Pertumbuhan Ekonomi	X8
Pengangguran Terbuka	X9

Kolom Daerah dihapus karena kolom daerah tidak berisikan tipe data numerik sehingga tidak diperlukan dalam perhitungan.

5.1.3 Matriks Korelasi

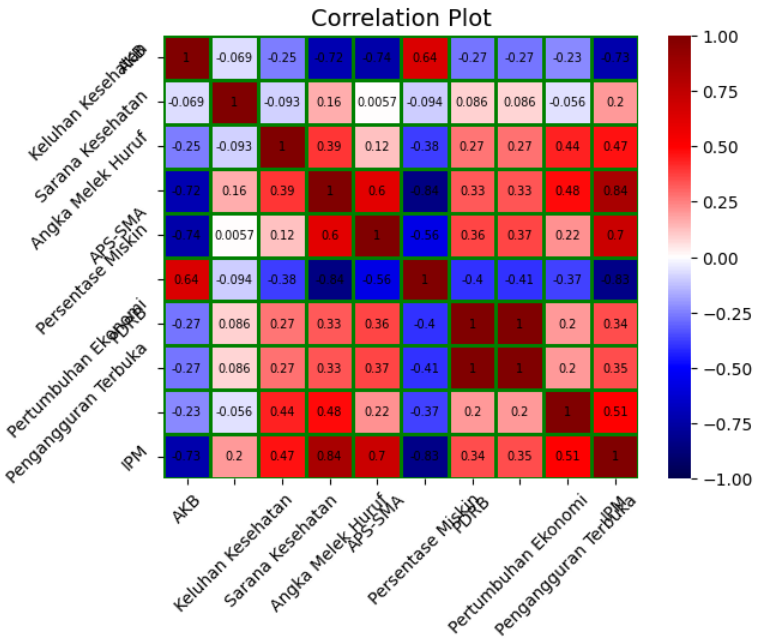
Menentukan matriks korelasi dari semua variabel atau fitur dari dataset yang digunakan

```
corr_matrix = kw.corr()
corr_matrix
```

✓ 0.0s

	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka	IPM
AKB	1.000000	-0.116594	-0.258732	-0.763542	-0.664867	0.623447	-0.552713	-0.570343	-0.113557	-0.731380
Keluhan Kesehatan	-0.116594	1.000000	0.153602	0.132343	-0.062334	-0.130683	0.206405	0.165364	0.088186	0.149093
Sarana Kesehatan	-0.258732	0.153602	1.000000	0.424559	0.062574	-0.393216	0.424723	0.411763	0.493145	0.502272
Angka Melek Huruf	-0.763542	0.132343	0.424559	1.000000	0.634086	-0.845115	0.719969	0.746246	0.415713	0.871347
APS-SMA	-0.664867	-0.062334	0.062574	0.634086	1.000000	-0.536298	0.427615	0.445413	0.156877	0.689825
Persentase Miskin	0.623447	-0.130683	-0.393216	-0.845115	-0.536298	1.000000	-0.802915	-0.824902	-0.324321	-0.819966
PDRB	-0.552713	0.206405	0.424723	0.719969	0.427615	-0.802915	1.000000	0.991172	0.286285	0.704331
Pertumbuhan Ekonomi	-0.570343	0.165364	0.411763	0.746246	0.445413	-0.824902	0.991172	1.000000	0.271336	0.725403
Pengangguran Terbuka	-0.113557	0.088186	0.493145	0.415713	0.156877	-0.324321	0.286285	0.271336	1.000000	0.427995
IPM	-0.731380	0.149093	0.502272	0.871347	0.689825	-0.819966	0.704331	0.725403	0.427995	1.000000

```
sns.heatmap(corr_matrix,
            cmap = 'seismic',
            linewidth = 0.75,
            linecolor = 'green',
            cbar = True,
            vmin = -1,
            vmax = 1,
            annot = True,
            annot_kws = {'size': 7,'color':'black'})
plt.tick_params(labelsize = 10, rotation = 45)
plt.title('Correlation Plot',size = 14)
```



Dari output yang dihasilkan dapat disimpulkan bahwa nilai korelasi yang cenderung warna merah berarti cukup tinggi(korelasi positif). Warna biru juga mempunyai korelasi yang cukup tinggi (korelasi negatif). Nilai korelasi diatas 0,5 merupakan nilai korelasi yang paling bagus. Akan tetapi apabila mendekati 1 harus dicek ulang hubungan antara variabel independent karena takutnya apabila sudah mendekati 1 akan terjasi multikolinearitas.

5.1.4 Inisiasi Variabel Responden & Prediktor

```
y = df_clean['IPM']  # Target variable
X = df_clean.drop(['IPM'], axis=1)  # Feature matrix
X.head()
```

✓ 0.0s

	AKB	Keluhan Kesehatan	Sarana Kesehatan	Angka Melek Huruf	APS-SMA	Persentase Miskin	PDRB	Pertumbuhan Ekonomi	Pengangguran Terbuka
0	19.71	42.07	3	92.57	76.12	14.19	5.51	5.47	1.39
1	22.21	53.78	5	89.11	77.50	10.36	5.31	5.27	3.77
2	18.28	48.01	1	94.41	76.78	12.02	5.03	5.03	4.12
3	18.91	48.32	12	96.84	78.54	7.27	5.21	5.21	2.53
4	20.67	41.78	7	94.49	68.80	9.72	5.10	5.10	3.38

Dari percobaan diatas didapatkan bahwa Variabel Responden (Y) adalah kolom IPM dan Variabel Prediktor (X) adalah kolom AKB, Keluhan Kesehatan, Angka Melek Huruf, APS-SMA, Persentase Miskin, PDRB, Pertumbuhan Ekonomi, Pengangguran Terbuka. Dari dataset ini dapat dianalisis seberapa pengaruhnya variabel prediktor(X) terhadap variabel responden(Y).

5.1.5 Regresi Linier

```
def processsubset(featureset):
    model = sm.OLS(y,X[list(featureset)])
    regr = model.fit()
    RSS = ((regr.predict(X[list(featureset)])-y)**2).sum()
    return {"model":regr,"RSS":RSS}
```

✓ 0.0s

Fungsi ini berguna untuk mengevaluasi kombinasi variabel prediktor dalam regresi linear. Nilai RSS yang lebih kecil menunjukkan model yang lebih baik dalam memprediksi data. Hasil dari variabel model dapat digunakan untuk analisis lebih lanjut seperti uji signifikansi, R-squared, dan interpretasi koefisien regresi.

```
def getbest(k):
    tic = time.time()
    results = []
    for combo in itertools.combinations(X.columns,k):
        results.append(processsubset(combo))
    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print("Processed", models.shape[0], k, "predictors in", toc-tic, "seconds")

    return best_model
```

Fungsi getbest(k) berguna untuk mencari kombinasi fitur terbaik dari ukuran k berdasarkan RSS terkecil. Digunakan dalam metode seleksi fitur untuk menentukan kombinasi optimal dalam regresi linear. Hasilnya bisa dianalisis lebih lanjut untuk interpretasi koefisien dan signifikansi statistik.

```
model_best = pd.DataFrame(columns = ["RSS","model"])

tic = time.time()
for i in range(1,10):
    model_best.loc[i] = getbest(i)
toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

[12] ✓ 0.7s

```
Processed 9 1 predictors in 0.04287862777709961 seconds
Processed 36 2 predictors in 0.04738879203796387 seconds
Processed 84 3 predictors in 0.11214065551757812 seconds
Processed 126 4 predictors in 0.17946648597717285 seconds
Processed 126 5 predictors in 0.1607956886291504 seconds
Processed 84 6 predictors in 0.12039995193481445 seconds
Processed 36 7 predictors in 0.05829334259033203 seconds
Processed 9 8 predictors in 0.011827945709228516 seconds
Processed 1 9 predictors in 0.0021076202392578125 seconds
Total elapsed time : 0.7525246143341064 seconds
```

Fungsi ini melakukan pemilihan kombinasi terbaik dari 1 hingga 9 variabel prediktor dalam regresi linier menggunakan RSS. Dari percobaan ini, didapatkan bahwa semakin banyak prediktor, semakin banyak kombinasi

```
model_best
✓ 0.0s
```

	RSS	model
1	250.637703	<statsmodels.regression.linear_model.Regressio...
2	211.631853	<statsmodels.regression.linear_model.Regressio...
3	169.75224	<statsmodels.regression.linear_model.Regressio...
4	163.873348	<statsmodels.regression.linear_model.Regressio...
5	159.403435	<statsmodels.regression.linear_model.Regressio...
6	158.080922	<statsmodels.regression.linear_model.Regressio...
7	156.730123	<statsmodels.regression.linear_model.Regressio...
8	155.802209	<statsmodels.regression.linear_model.Regressio...
9	155.732593	<statsmodels.regression.linear_model.Regressio...

Variabel model_best menampilkan model regresi terbaik berdasarkan Residual Sum of Squares (RSS) untuk setiap jumlah prediktor dari 1 hingga 9. Semakin banyak prediktor, nilai RSS semakin kecil

A. Permodelan Regresi Linier dengan metode Ordinary Least Squares (OLS)

OLS Regression Results

Dep. Variable:	IPM	R-squared (uncentered):	0.999
Model:	OLS	Adj. R-squared (uncentered):	0.999
Method:	Least Squares	F-statistic:	1.635e+04
Date:	Mon, 17 Mar 2025	Prob (F-statistic):	5.56e-54
Time:	22:14:27	Log-Likelihood:	-86.548
No. Observations:	38	AIC:	177.1
Df Residuals:	36	BIC:	180.4
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Angka Melek Huruf	0.6697	0.038	17.519	0.000	0.592	0.747
APS-SMA	0.1209	0.047	2.576	0.014	0.026	0.216

Omnibus:	10.075	Durbin-Watson:	1.567
Prob(Omnibus):	0.006	Jarque-Bera (JB):	9.602
Skew:	0.943	Prob(JB):	0.00822
Kurtosis:	4.584	Cond. No.	18.3

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Metode OLS ini untuk memodelkan variabel IPM berdasarkan dua prediktor yakni pada Angka Melek Huruf dan APS-SMA. Hasil regresi menyatakan bahwa nilai R-squared mencapai 0.999 artinya model menjelaskan sebesar 99.9% variabilitas dalam IPM, serta terdapat nilai adj. R-squared sebesar 0.99 setelah penyesuaian terhadap jumlah prediktor model tetap sangat baik. Selanjutnya terdapat nilai P-value pada kedua variabel. Variabel X4 (Angka Melek Huruf) mempunyai nilai sebesar 0.000 artinya Sangat kecil (di bawah 0.05), menunjukkan bahwa variabel ini sangat signifikan dalam mempengaruhi IPM. Lalu pada variabel (X5) APS-SMA menunjukkan nilai P-value Juga lebih kecil dari 0.05, sehingga signifikan dalam model regresi.

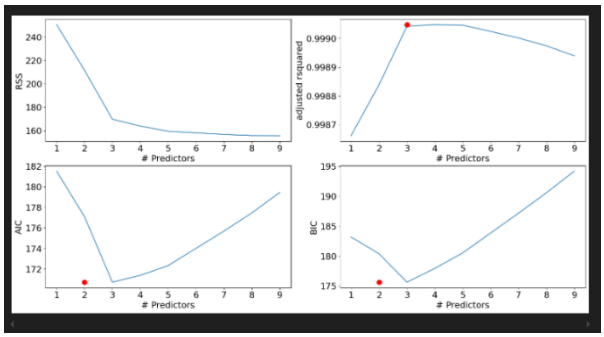
```
print(model_best.loc[2,"model"].rsquared)
✓ 0.0s
0.9989000501154163
```

```
model_best.apply(lambda row:row[1].rsquared, axis=1)
✓ 0.0s
1    0.998697
2    0.998900
3    0.999118
4    0.999148
5    0.999172
6    0.999178
7    0.999185
8    0.999190
9    0.999191
dtype: float64
```

Dari Percobaan ini berarti bahwa hampir 99.86% atau lebih variabilitas dalam data dapat dijelaskan oleh model, yang menunjukkan model sangat baik dalam menjelaskan hubungan antar variabel. Nilai R^2 meningkat dengan bertambahnya jumlah predictor Awalnya $R^2 = 0.9989$

C. Menampilkan Plot untuk mengevaluasi performa model regresi

```
plt.figure(figsize=(20,10))
plt.rcParams.update({'font.size':18,'lines.markersize':10})

plt.subplot(2,2,1)

plt.plot(model_best["RSS"])
plt.xlabel("# Predictors")
plt.ylabel("RSS")

rsquared_adj = model_best.apply(lambda row: row[1].rsquared_adj, axis=1)

plt.subplot(2,2,2)
plt.plot(rsquared_adj)
plt.plot(rsquared_adj.argmax(), rsquared_adj.max(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('adjusted rsquared')

aic = model_best.apply(lambda row: row[1].aic, axis=1)

plt.subplot(2,2,3)
plt.plot(aic)
plt.plot(aic.argmin(), aic.min(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('AIC')

bic = model_best.apply(lambda row: row[1].bic, axis=1)

plt.subplot(2,2,4)
plt.plot(bic)
plt.plot(bic.argmin(), bic.min(), 'or')
plt.xlabel("# Predictors")
plt.ylabel('BIC')
```



Model dengan **3 prediktor** adalah yang optimal berdasarkan semua metrik evaluasi. Menambah lebih banyak prediktor setelah titik ini hanya meningkatkan kompleksitas tanpa peningkatan prediksi yang signifikan. Model dengan terlalu banyak prediktor dapat menyebabkan overfitting, sehingga memilih jumlah prediktor yang lebih sedikit namun informatif lebih baik.

D. Pemilihan Metode Subset terbaik

1. Forward Selection

   Forward Selection (Seleksi Maju), Metode ini memulai proses dengan model tanpa variabel prediktor. Selanjutnya, variabel prediktor ditambahkan satu per satu ke dalam model berdasarkan tingkat signifikansinya, biasanya menggunakan nilai p-value. Variabel dengan p-value paling kecil ditambahkan terlebih dahulu. Proses ini berlanjut hingga tidak ada lagi variabel yang memenuhi kriteria signifikansi untuk dimasukkan ke dalam model. Penentuan nilai forward selection dapat menggunakan source code dibawah ini

INPUT OUTPUT

```
def forward(predictors):
    remaining_predictors = [p for p in X.columns if p not in predictors]

    tic = time.time()

    results = []

    for p in remaining_predictors:
        results.append(processsubset(predictors+[p]))

    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()

    print("Processed", models.shape[0], "predictors in", toc-tic, "seconds")

    return best_model
0.0s
```

```
models_fwd = pd.DataFrame(columns = ["RSS","model"])

tic = time.time()
predictors = []

for i in range(1, len(X.columns)+1):
    models_fwd.loc[i] = forward(predictors)
    predictors = models_fwd.loc[i]["model"].model.exog_names

toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

```
Processed 9 predictors in 0.018275022506713867 seconds
Processed 8 predictors in 0.010036468505859375 seconds
Processed 7 predictors in 0.008873462677001953 seconds
Processed 6 predictors in 0.00816202163696289 seconds
Processed 5 predictors in 0.007517099380493164 seconds
Processed 4 predictors in 0.006398677825927734 seconds
Processed 3 predictors in 0.01107931137084961 seconds
Processed 2 predictors in 0.002179384231567383 seconds
Processed 1 predictors in 0.002513408660888672 seconds
Total elapsed time : 0.0849156379699707 seconds
```

2. Backward Selection

Metode ini dimulai dengan model yang mencakup semua variabel prediktor yang tersedia. Kemudian, variabel yang paling tidak signifikan (biasanya dengan p-value terbesar) dihapus dari model. Proses penghapusan ini berlanjut secara iteratif hingga semua variabel yang tersisa dalam model memiliki tingkat signifikansi yang memenuhi kriteria yang ditetapkan

INPUT

```
def backward(predictors):
    tic = time.time()

    results = []

    for combo in itertools.combinations(predictors,len(predictors)-1):
        results.append(processsubset(combo))
    models = pd.DataFrame(results)

    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print('Processed',models.shape[0],'models on',len(predictors)-1, 'predictors in',toc-tic,'seconds')

    return best_model
0.0s
```

```
models_bwd = pd.DataFrame(columns = ["RSS","model"], index = range(1,len(X.columns)))

tic = time.time()
predictors = X.columns

while(len(predictors) > 1):
    models_bwd.loc[len(predictors)-1] = backward(predictors)
    predictors = models_bwd.loc[len(predictors)-1]['model'].model.exog_names
toc = time.time()
print("Total elapsed time :",(toc-tic),"seconds")
```

OUTPUT

```
Processed 9 models on 8 predictors in 0.02596569061279297 seconds
Processed 8 models on 7 predictors in 0.019109487533569336 seconds
Processed 7 models on 6 predictors in 0.009807109832763672 seconds
Processed 6 models on 5 predictors in 0.006870269775390625 seconds
Processed 5 models on 4 predictors in 0.008771657943725586 seconds
Processed 4 models on 3 predictors in 0.006197690963745117 seconds
Processed 3 models on 2 predictors in 0.00591063499450683 seconds
Processed 2 models on 1 predictors in 0.002073526382446289 seconds
Total elapsed time : 0.08990669250488281 seconds
```

Dari output yang dihasilkan menunjukkan bahwa Ketika jumlah prediktor menurun maka waktu eksekusi cenderung berkurang, contohnya pada pemrosesan 9 model dengan menggunakan 8 prediktor membutuhkan waktu 0.02596 detik sementara 2 model dengan 1 prediktor hanya membutuhkan 0.0020735 detik.

3. Perbandingan Hasil

```
print("FORWARD SELECTION")
print(models_fwd.loc[7,'model'].params)
0.0s
```

```
FORWARD SELECTION
Angka Melek Huruf       0.539865
APS-SMA                 0.148162
Sarana Kesehatan        0.116435
Pertumbuhan Ekonomi     3.921060
Keluhan Kesehatan       0.067926
PDRB                   -2.974397
Pengangguran Terbuka    0.195383
dtype: float64
```

```
print("BACKWARD SELECTION")
print(models_bwd.loc[7,'model'].params)
0.0s
```

```
BACKWARD SELECTION
Keluhan Kesehatan       0.067926
Sarana Kesehatan        0.116435
Angka Melek Huruf       0.539865
APS-SMA                 0.148162
PDRB                   -2.974397
Pertumbuhan Ekonomi     3.921060
Pengangguran Terbuka    0.195383
dtype: float64
```

Forward Selection	Backward Selection
Model dimulai dengan tidak ada rediktor, kemudian model menambahkan variabel satu per satu yang paling meningkatkan performa model	Model dimulai dengan semua prediktor, lalu menghapus satu per satu variabel yang paling tidak signifikan terhadap model

Model yang terbentuk melalui Forward dan Backward Selection cukup konsisten, dengan variabel yang dipilih hampir sama. **Pertumbuhan Ekonomi** adalah prediktor terkuat dengan dampak positif terbesar. **PDRB** berpengaruh negatif terhadap target. Variabel lain memiliki pengaruh lebih kecil tetapi tetap signifikan. Hasil ini dapat digunakan untuk memahami faktor-faktor utama yang mempengaruhi target dalam analisis.

5.1.6 Model Regresi Ridge

A. Menghapus baris naN

INPUT

```
df = pd.read_excel('IPM JATIM 2018.xlsx').dropna().drop('Daerah', axis = 1)
✓ 0.0s

y = df['IPM']
X = df.drop(['IPM'], axis = 1).astype('float64')
✓ 0.0s

alphas = 10**np.linspace(10,-2,100)*0.5
alphas
```

OUTPUT

```
array([5.00000000e+09, 3.78231664e+09, 2.86118383e+09, 2.16438064e+09,
       1.63727458e+09, 1.23853818e+09, 9.36908711e+08, 7.08737081e+08,
       5.36133611e+08, 4.05565415e+08, 3.06795364e+08, 2.32079442e+08,
       1.75559587e+08, 1.32804389e+08, 1.00461650e+08, 7.59955541e+07,
       5.74878498e+07, 4.34874501e+07, 3.28966612e+07, 2.48851178e+07,
       1.88246790e+07, 1.42401793e+07, 1.07721735e+07, 8.14875417e+06,
       6.16423370e+06, 4.66301673e+06, 3.52740116e+06, 2.66834962e+06,
       2.01850863e+06, 1.52692775e+06, 1.15506485e+06, 8.73764200e+05,
       6.60970574e+05, 5.00000000e+05, 3.78231664e+05, 2.86118383e+05,
       2.16438064e+05, 1.63727458e+05, 1.23853818e+05, 9.36908711e+04,
       7.08737081e+04, 5.36133611e+04, 4.05565415e+04, 3.06795364e+04,
       2.32079442e+04, 1.75559587e+04, 1.32804389e+04, 1.00461650e+04,
       7.59955541e+03, 5.74878498e+03, 4.34874501e+03, 3.28966612e+03,
       2.48851178e+03, 1.88246790e+03, 1.42401793e+03, 1.07721735e+03,
       8.14875417e+02, 6.16423370e+02, 4.66301673e+02, 3.52740116e+02,
       2.66834962e+02, 2.01850863e+02, 1.52692775e+02, 1.15506485e+02,
       8.73764200e+01, 6.60970574e+01, 5.00000000e+01, 3.78231664e+01,
       2.86118383e+01, 2.16438064e+01, 1.63727458e+01, 1.23853818e+01,
       9.36908711e+00, 7.08737081e+00, 5.36133611e+00, 4.05565415e+00,
       3.06795364e+00, 2.32079442e+00, 1.75559587e+00, 1.32804389e+00,
       1.00461650e+00, 7.59955541e-01, 5.74878498e-01, 4.34874501e-01,
       3.28966612e-01, 2.48851178e-01, 1.88246790e-01, 1.42401793e-01,
       1.07721735e-01, 8.14875417e-02, 6.16423370e-02, 4.66301673e-02,
       3.52740116e-02, 2.66834962e-02, 2.01850863e-02, 1.52692775e-02,
       1.15506485e-02, 8.73764200e-03, 6.60970574e-03, 5.00000000e-03])
```

Interpretasi Hasil

Perintah diatas melakukan pemrosesan dataset IPM JATIM 2018 dengan menghapus baris dengan nilai kosong lalu menghapus kolom “Daerah” karena kolom tersebut tidak dapat digunakan perhitungan secara numerik. Variable dependen (y) diambil dari kolom IPM dan variable independenc(x) merupakan seluruh kolom selain IPM yang nantinya akan dikonversikan kedalam float untuk memastikan tipe data numerik. Disini saya membuat parameter 100 angka yang teristribusi merata antara 10 hingga -2 yang nantinya diubah kedalam skala algoritma dari 10^{10} hingga 10^{-2}

B. Pengujian Dengan Nilai Alpha

INPUT & OUTPUT

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # Normalisasi fitur
✓ 0.0s

ridge = Ridge()
coefs = []

for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(X_scaled, y)
    coefs.append(ridge.coef_)

np.shape(coefs)
✓ 0.2s

(100, 9)
```

Interpretasi Hasil

Pemrosesan ini menguji Ridge Regression pada berbagai nilai alpha untuk melihat bagaimana regularisasi mempengaruhi koefisien regresi. Koefisien akan berubah seiring perubahan alpha, menunjukkan efek regularisasi terhadap model. Alpha mengontrol besarnya penalti L2 dalam Ridge Regression. Jika alpha kecil, model mendekati regresi linear biasa. Jika alpha besar, koefisien regresi mengecil, bahkan mendekati nol.

C . Implementasi Nilai Alpha pada Plot

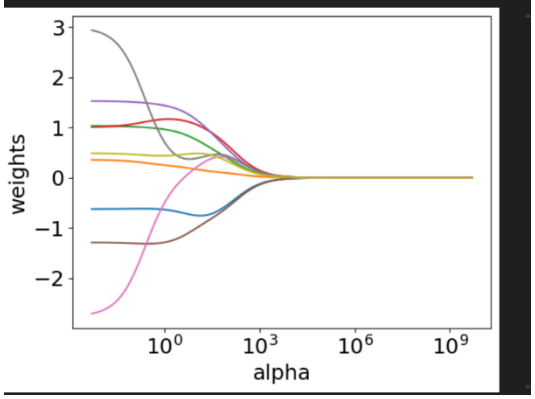
INPUT & OUTPUT

“Pembuatan Plot”

```
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

0.2s

```
xt(0, 0.5, 'weights')
```



Interpretasi Hasil

Plot disamping menunjukkan bagaimana koefisien regresi ridge berubah terhadap nilai alpha dalam skala logaritmik. Sumbu X merupakan nilai alpha dalam Regresi Ridge. Jika nilai alpha kecil maka koefisien semakin besar dan juga jika alpha besar maka semakin besar penalty L2 yang membuat koefisien semakin kecil

D . Partisi dataset training dan testing

```
X_train, X_test , y_train, y_test = train_test_split(X, y,test_size=0.5, random_state=1)

# Normalisasi hanya berdasarkan X_train
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[33] 0.0s

Melakukan pemisahan dataset training dan dataset testing karena agar terhindar dari overfitting. Overfitting terjadi saat model terlalu kompleks dan menangkap noise dalam data training, sehingga kinerjanya buruk saat diuji pada data baru. Selain itu pemisahan ini juga digunakan untuk mengevaluasi kinerja model secara objektif menggunakan Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared (R^2).

E. Interpretasi Hasil Regresi Ridge dengan alpha = 4

```
ridge2 = Ridge(alpha=4)  # Hapus normalize=True
ridge2.fit(X_train_scaled, y_train)
pred2 = ridge2.predict(X_test_scaled)

print(pd.Series(ridge2.coef_, index = X.columns))
print(mean_squared_error(y_test, pred2))
```

0.0s

```
AKB                    -0.607453
Keluhan Kesehatan       0.607473
Sarana Kesehatan        1.058454
Angka Melek Huruf       0.936433
APS-SMA                 0.711016
Persentase Miskin      -0.219814
PDRB                    0.251129
Pertumbuhan Ekonomi     0.510900
Pengangguran Terbuka    0.526140
dtype: float64
7.707132586132113
```

Hasil ini menunjukkan koefisien regresi Ridge dan Mean Squared Error (MSE) dari model dengan alpha = 4. Setiap variabel memiliki nilai koefisien yang menunjukkan pengaruhnya terhadap target (y). Koefisien positif, variabel berkorelasi positif dengan target(peningkatan variabel ini meningkatkan target). Koefisien negatif, variabel berkorelasi negatif dengan target (peningkatan variabel ini menurunkan target).

Sarana Kesehatan 1.058454, berpengaruh paling besar secara positif terhadap target. Angka Melek Huruf 0.936433 , juga memiliki dampak besar dan positif. AKB -0.607453 memiliki dampak negatif, artinya semakin tinggi AKB, semakin rendah target. Persentase Miskin -0.219814 juga berdampak negatif.

```
ridge3 = Ridge(alpha=10**10)  # Hapus normalize=True
ridge3.fit(X_train_scaled, y_train)
pred3 = ridge3.predict(X_test_scaled)

print(pd.Series(ridge3.coef_, index=X.columns))

# Hitung Mean Squared Error (MSE)
print(mean_squared_error(y_test, pred3))
```

✓ 0.0s

```
AKB                    -4.564013e-09
Keluhan Kesehatan       1.814386e-09
Sarana Kesehatan        4.284484e-09
Angka Melek Huruf       6.068905e-09
APS-SMA                 3.079861e-09
Persentase Miskin      -4.910514e-09
PDRB                    5.098921e-09
Pertumbuhan Ekonomi     5.161990e-09
Pengangguran Terbuka    3.609187e-09
dtype: float64
37.48226187503708
```

F. Mengecek Alpha Terbaik dengan RidgeCV

```
ridgecv = RidgeCV(alphas=alphas, scoring='neg_mean_squared_error')
ridgecv.fit(X_train_scaled, y_train)

# Cek alpha terbaik
ridgecv.alpha_
```

✓ 0.0s

```
np.float64(21.643806405415308)
```

```
ridge4 = Ridge(alpha=ridgecv.alpha_)  # Hapus normalize=True
ridge4.fit(X_train_scaled, y_train)

# Hitung MSE pada data test
mse = mean_squared_error(y_test, ridge4.predict(X_test_scaled))
print("Mean Squared Error:", mse)
```

✓ 0.0s

```
Mean Squared Error: 9.392908657163696
```

```
ridge4.fit(X, y)
pd.Series(ridge4.coef_, index = X.columns)
```

✓ 0.0s

```
AKB                    -0.049795
Keluhan Kesehatan       0.038137
Sarana Kesehatan        0.136349
Angka Melek Huruf       0.253247
APS-SMA                 0.144626
Persentase Miskin      -0.303898
PDRB                    0.024181
Pertumbuhan Ekonomi     0.060959
Pengangguran Terbuka    0.217480
```

Alpha terbaik ditemukan sebesar 21.64 ditemukan menggunakan RidgeCV, Nilai MSE sebesar 9.39 cukup rendah, yang menunjukkan bahwa model ini memiliki performa yang cukup baik. Koefisien regresi menunjukkan dampak masing-masing variabel prediktor terhadap variabel target:

1. Keluhan Kesehatan 0.3812 dan Pengangguran Terbuka 0.2175 memiliki pengaruh positif yang cukup besar terhadap variabel target.
2. APS-SMA 0.1446 juga memiliki pengaruh positif, meskipun lebih kecil.
3. Persentase Miskin 0.3039 juga berkontribusi positif.
4. Beberapa variabel seperti AKB -0.0498 memiliki pengaruh negatif kecil.

5.1.7 Model Regresi Lasso

A. Interpretasi Hasil

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

✓ 0.0s

```
# Lasso Regression tanpa normalize=True
lasso = Lasso(max_iter=10000)

coefs = []
for a in alphas:
    lasso.set_params(alpha=a)
    lasso.fit(X_train_scaled, y_train)
    coefs.append(lasso.coef_)
```

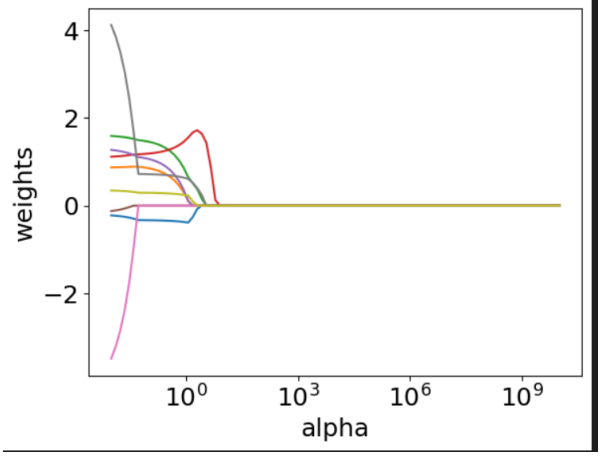
Pemrosesan ini menguji Lasso Regression pada berbagai nilai alpha untuk melihat bagaimana regularisasi mempengaruhi koefisien regresi. Lasso Regression digunakan untuk melakukan seleksi fitur dengan L1 regularization. Hasil alpha dalam Lasso Regression menentukan seberapa banyak fitur yang dipertahankan atau dihapus.

B. Implementasi Nilai Alpha pada Plot

```
ax = plt.gca()
ax.plot(alphas*2, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

✓ 0.1s

```
ext(0, 0.5, 'weights')
```



Interpretasi Hasil

Plot disamping menunjukkan bagaimana koefisien regresi lasso berubah terhadap nilai alpha dalam skala logaritmik. Sumbu X merupakan nilai alpha dalam Regresi Lasso. Terdapat Alpha optimal dengan nilai di mana sebagian besar fitur yang tidak relevan dihapus, tetapi model masih mempertahankan informasi penting. Nilai alpha terlalu kecil menyebabkan overfitting dengan semua fitur dipertahankan. Serta terdapat Nilai alpha terlalu besar dan mengakibatkan underfitting yang nantinya semua fitur dihapus

5.1.8 Mengecek Alpha Terbaik dengan LassoCV

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

✓ 0.0s

```
lassocv = LassoCV(alphas=None, cv=10, max_iter=100000)
lassocv.fit(X_train_scaled, y_train)

# Gunakan alpha terbaik dari LassoCV
lasso = Lasso(alpha=lassocv.alpha_, max_iter=100000)
lasso.fit(X_train_scaled, y_train)

# Hitung MSE pada data test
mse = mean_squared_error(y_test, lasso.predict(X_test_scaled))
print("Mean Squared Error:", mse)
```

✓ 0.0s

```
Mean Squared Error: 10.357918335224445
```

```
pd.Series(lasso.coef_, index=X.columns)
```

✓ 0.0s

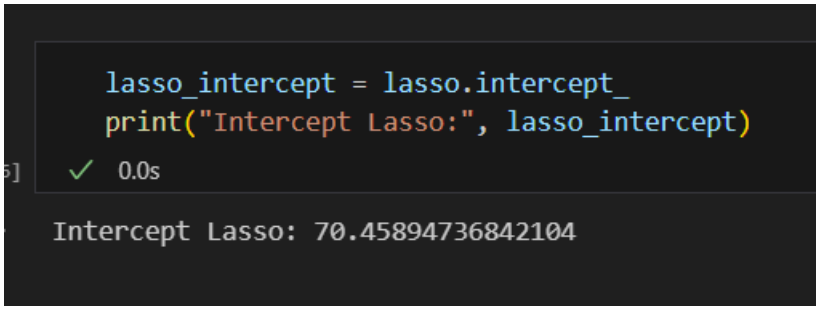
```
AKB                    -0.173492
Keluhan Kesehatan       0.000000
Sarana Kesehatan        0.482793
Angka Melek Huruf       1.722500
APS-SMA                 0.000000
Persentase Miskin      -0.000000
PDRB                    0.000000
Pertumbuhan Ekonomi     0.467420
Pengangguran Terbuka    0.007631
```

Interpretasi Hasil

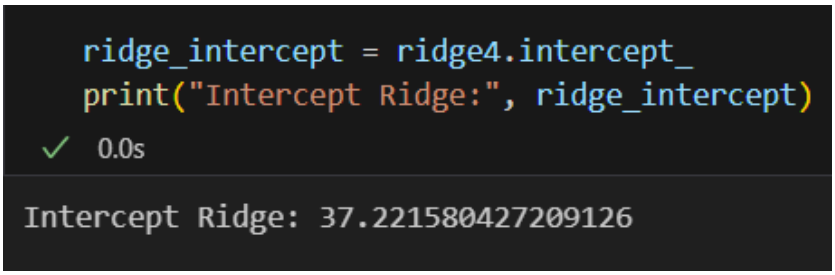
Nilai MSE sebesar 10.35791 menunjukkan seberapa besar kesalahan prediksi model dibandingkan dengan nilai aktual. Lasso dapat menghapus fitur yang tidak signifikan seperti Keluhan Kesehatan, APS-SMA, Persentase Miskin, PDRB Fitur yang tersisa lebih relevan.

Mengecek Intercept Model

A. Lasso Regresion



B. Ridge Regresion



Interpretasi Hasil Terbaik

Ridge Regression adalah model yang lebih baik karena memiliki MSE yang jauh lebih rendah sebesar 37.2215 dibandingkan Lasso sebesar 70.458. hal Ini menunjukkan bahwa Ridge menghasilkan prediksi yang lebih akurat dalam pemrosesan dataset ini