LAPORAN PRAKTIKUM
PENERAPAN MACHINE LEARNING
MENGGUNAKAN ALGORITMA REGRESI LINIER
MATA KULIAH MACHINE LEARNING
KELAS C

DISUSUN OLEH:

SANDRIA AMELIA PUTRI

21083010005

DOSEN PENGAMPU:

AVIOLLA TERZA DAMALIANA, S.SI., M.STAT

DR. ENG. IR. ANGGRAINI PUSPITA SARI., ST., MT.

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2023

DAFTAR ISI

Contents

BAB 1: PENDAHULUAN

1.1. Latar Belakang

Statistika dan machine learning membentuk satu konsep yang disebut statistical machine learning dengan menggunakan model regresi logistik. Statistika dan machine learning menggunakan teori serta rumus yang sama dalam menyelesaikan suatu permasalahan. Keduanya sama-sama membahas konsep random variable, distribusi statistik, expected value, variansi, sampai pada konsep distribusi prior dan posterior (R., 2018).

Analisis regresi linier merupakan sebuah metode untuk memprediksi masa depan yang didasarkan oleh perhitungan matematika, yakni dengan memperhitungkan variabel yang ada. Selain itu, juga bertujuan untuk memeriksa apakah variabel independen (prediktor) mampu memprediksi variabel dependen (hasil) serta variabel mana yang memberi dampak signifikan terhadap hasil prediksi (Algoritma, 2022). Pada laporan kali ini akan dilakukan analisis regresi menggunakan algoritma machine learning pada suatu permasalahan.

1.2. Permasalahan

Untuk penerapan machine learning menggunakan algoritma regresi linier digunakan dataset “The Boston Housing”. Permasalahannya yaitu ingin memprediksi value rata-rata rumah ($1000) dari dataset tersebut. Dataset ini merupakan kumpulan data yang berasal dari informasi yang dikumpulkan oleh layanan sensus Amerika Serikat mengenai perumahan di wilayah Boston MA. Berikut merupakan bentuk data dalam excel:

	A	B	C	D	E	F	G	H	I
1	0.00632	18.00	2.310	0	0.5380	6.5750	65.20	4.0900	1	296.0	15.30	396.90	4.98	24.00
2	0.02731	0.00	7.070	0	0.4690	6.4210	78.90	4.9671	2	242.0	17.80	396.90	9.14	21.60
3	0.02729	0.00	7.070	0	0.4690	7.1850	61.10	4.9671	2	242.0	17.80	392.83	4.03	34.70
4	0.03237	0.00	2.180	0	0.4580	6.9980	45.80	6.0622	3	222.0	18.70	394.63	2.94	33.40
5	0.06905	0.00	2.180	0	0.4580	7.1470	54.20	6.0622	3	222.0	18.70	396.90	5.33	36.20
6	0.02985	0.00	2.180	0	0.4580	6.4300	58.70	6.0622	3	222.0	18.70	394.12	5.21	28.70
7	0.08829	12.50	7.870	0	0.5240	6.0120	66.60	5.5605	5	311.0	15.20	395.60	12.43	22.90
8	0.14455	12.50	7.870	0	0.5240	6.1720	96.10	5.9505	5	311.0	15.20	396.90	19.15	27.10
9	0.21124	12.50	7.870	0	0.5240	5.6310	100.00	6.0821	5	311.0	15.20	386.63	29.93	16.50
10	0.17004	12.50	7.870	0	0.5240	6.0040	85.90	6.5921	5	311.0	15.20	386.71	17.10	18.90
11	0.22489	12.50	7.870	0	0.5240	6.3770	94.30	6.3467	5	311.0	15.20	392.52	20.45	15.00
12	0.11747	12.50	7.870	0	0.5240	6.0090	82.90	6.2267	5	311.0	15.20	396.90	13.27	18.90
13	0.09378	12.50	7.870	0	0.5240	5.8890	39.00	5.4509	5	311.0	15.20	390.50	15.71	21.70
14	0.62976	0.00	8.140	0	0.5380	5.9490	61.80	4.7075	4	307.0	21.00	396.90	8.26	20.40
15	0.63796	0.00	8.140	0	0.5380	6.0960	84.50	4.4619	4	307.0	21.00	380.02	10.26	18.20
16	0.62739	0.00	8.140	0	0.5380	5.8340	56.50	4.4986	4	307.0	21.00	395.62	8.47	19.90
17	1.05393	0.00	8.140	0	0.5380	5.9350	29.30	4.4986	4	307.0	21.00	386.85	6.58	23.10
18	0.78420	0.00	8.140	0	0.5380	5.9900	81.70	4.2579	4	307.0	21.00	386.75	14.67	17.50
19	0.80271	0.00	8.140	0	0.5380	5.4560	36.60	3.7965	4	307.0	21.00	288.99	11.69	20.20

Berikut merupakan deskripsi data:

No	Variabel	Keterangan	Jenis Data
1	CRIM	Tingkat kejahatan per kapita menurut kota	Numerik
2	ZN	Proporsi lahan perumahan yang dizonasikan untuk kavling lebih dari 25.000 kaki persegi	Numerik
3	INDUS	Proporsi hektar bisnis non-ritel per kota	Numerik
4	CHAS	Variabel dummy Sungai Charles (1 jika saluran membatasi sungai; 0 jika tidak)	Kategorik
5	NOX	Konsentrasi oksida nitrat	Numerik

6	RM	Jumlah kamar rata-rata tempat tinggal	Numerik
7	AGE	Proporsi unit yang ditempati pemilik yang dibangun sebelum tahun 1940	Numerik
8	DIS	Pembobot jarak ke lima pusat pekerja Boston	Numerik
9	RAD	Indeks aksesibilitas ke jalan raya radial	Kategorik
10	TAX	Tarif pajak properti nilai penuh per $10.000	Numerik
11	PTARATIO	Rasio murid-guru menurut kota	Numerik
12	B	Proporsi orang kulit hitam	Numerik
13	LSTAT	Proporsi populasi yang memiliki status rendah	Numerik
14	MEDV	Value rata-rata rumah ($1000)	Numerik

Variabel MEDV merupakan variabel target yang merupakan inti permasalahannya dan ingin diprediksi nilainya.

1.3. Tujuan

1. Memahami teori mengenai regresi linier yang mencakup regresi linier sederhana, regresi linier berganda, regresi polinomial, regresi ridge, dan regresi lasso.
2. Menerapkan algoritma regresi pada permasalahan machine learning.
3. Menganalisis hasil dari algoritma regresi pada permasalahan machine learning.

1.4. Manfaat

1. Dapat memahami teori mengenai regresi linier yang mencakup regresi linier sederhana, regresi linier berganda, regresi polinomial, regresi ridge, dan regresi lasso.
2. Dapat menerapkan algoritma regresi pada permasalahan machine learning.
3. Dapat menganalisis hasil dari algoritma regresi pada permasalahan machine learning.

BAB 2: TINJAUAN PUSTAKA

2.1. Regresi Linier

Metode regresi linier merupakan metode analisis yang paling banyak digunakan (Delyani, 2021). Regresi linier dapat digunakan ketika terdapat hubungan yang linier antara variabel independen dan variabel dependen (Maulid, 2022). Berikut merupakan bentuk persamaan umum sebuah model regresi linier:

$$y_i = a + b_i x_i$$

dengan,

y: nilai prediksi
a: konstanta (intersept)
x: variabel-variabel independen
b: koefisien regresi

Secara umum, model regresi linier dibagi menjadi dua jenis yaitu model regresi linier sederhana (melibatkan satu variabel dependen dengan satu variabel independen) dan regresi linier berganda (melibatkan satu variabel dependen dengan lebih dari satu variabel independen).

2.2. Regresi Polinomial

Model regresi polinomial dapat mengandung satu, dua atau lebih dari dua variabel bebas. Jika hanya terdapat satu variabel bebas, maka disebut model regresi polinomial sederhana dan jika terdapat dua atau lebih dari dua variabel bebas maka disebut model regresi polinomial berganda (Stat, 2020). Berikut merupakan bentuk persamaan umum sebuah model regresi polinomial sederhana orde 2:

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i$$

di mana $x_i = X_i - \bar{X}$

Bentuk persamaan di atas disebut orde 2 karena variabel bebas tunggal muncul dalam bentuk pangkat pertama dan kedua. Penentuan derajat polinomial dapat ditentukan berdasarkan kontras-kontras ortogonal yang nyata (signifikan) dari masing-masing faktor, kemudian dapat ditemukan bentuk regresi pendekatannya (Widiharih, 2001). Analisis regresi polinomial derajat dua berkelompok digunakan untuk memodelkan hubungan antara peubah respon dan peubah prediktor kuantitatif dan kualitatif di mana data cenderung membentuk kurva kuadratik (Setyabudi, 2017).

2.3. Regresi Ridge

Regresi ridge adalah suatu teknik yang dikembangkan untuk menstabilkan nilai koefisien regresi karena adanya multikolinieritas. Metode regresi ridge pertama kali dikemukakan oleh A. E. Hoerl pada tahun 1962. Hasil dari metode ini diperoleh melalui penambahan konstanta bias c pada diagonal XTX (Wasilaine, Talakua, & Lesnussa, 2014).

Regresi ridge merupakan modifikasi dari metode kuadrat terkecil yang menghasilkan penduga bias dari koefisien regresi. Regresi ini mengurangi dampak multikolinieritas dengan menentukan penduga yang bias tetapi mempunyai varians yang lebih kecil dari varians penduga regresi linier berganda (Wasilaine, Talakua, & Lesnussa, Model Regresi Ridge Untuk Mengatasi Model Regresi Linier Berganda Yang Mengandung Multikolinieritas, 2014). Berikut merupakan bentuk persamaan umum sebuah model regresi ridge:

$$\hat{\beta}(c) = (X^T X + cI)^{-1} X^T Y$$

2.4. Regresi Lasso

Regresi lasso (Least Absolute Shrinkage and Selection Operator) mirip dengan regresi ridge. Tipe analisis ini mampu mengurangi variabilitas dan meningkatkan akurasi model regresi linier. Perbedaannya yaitu pada regresi lasso menggunakan nilai absolut, bukan kuadrat seperti pada regresi ridge. Dengan begitu, nilai yang menyebabkan estimasi parameter bisa benar-benar nol (algoritma, 18). Berikut merupakan bentuk persamaan umum sebuah model regresi lasso (Robbani, Agustiani, & Herrhyanto, 2019):

$$\hat{\beta}^{LASSO} = argmin\left\{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right\}$$

BAB 3: ANALISIS DAN PEMBAHASAN

3.1. Menampilkan dataset The Boston Housing ke dalam dataFrame

- Import library yang diperlukan yaitu pandas dan numpy.

```
[1]: # import library
     import pandas as pd
     import numpy as np
```

- Tampilkan dataset The Boston Housing ke dalam dataFrame dengan menggunakan command pd.read_csv(). Simpan pada variabel baru yaitu housing.

```
[2]: # menampilkan dataset The Boston Housing ke dalam dataFrame
     housing = pd.read_csv(r"C:\Users\Sandria\Python\Semester 4\Machine Learning\housing.csv")
     housing
```

[2]:

	0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30 396.90 4.98 24.00
0	0.02731 0.00 7.070 0 0.4690 6.4210 78...
1	0.02729 0.00 7.070 0 0.4690 7.1850 61...
2	0.03237 0.00 2.180 0 0.4580 6.9980 45...
3	0.06905 0.00 2.180 0 0.4580 7.1470 54...
4	0.02985 0.00 2.180 0 0.4580 6.4300 58...
...	...
500	0.06263 0.00 11.930 0 0.5730 6.5930 69...
501	0.04527 0.00 11.930 0 0.5730 6.1200 76...
502	0.06076 0.00 11.930 0 0.5730 6.9760 91...
503	0.10959 0.00 11.930 0 0.5730 6.7940 89...
504	0.04741 0.00 11.930 0 0.5730 6.0300 80...

505 rows × 1 columns

- Karena dataset tersebut memiliki header yang tidak sesuai serta setiap kolomnya belum terpisah, maka berikan nama_variabel pada header dan pisah dataset menjadi beberapa kolom.

```
[3]: # memberikan nama_variabel pada header data
     Nama_Variabel = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LISTAT', 'MEDV']
     housing = pd.read_csv(r'housing.csv', header=None, delim_whitespace=True, names=Nama_Variabel)
     housing.head()
```

[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LISTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

3.2. Melakukan tahapan pre-processing data

- Periksa dimensi data dengan menggunakan command np.shape(). Dihasilkan output bahwa dataset tersebut memiliki 506 baris dan 14 kolom.

```
[4]: # memeriksa dimensi data
     np.shape(housing)
```

```
[4]: (506, 14)
```

- Periksa jenis data setiap variabel/kolom dengan menggunakan command dtypes. Dihasilkan output bahwa data tersebut ada yang bertipe float dan integer.

```
[5]: # memeriksa jenis data setiap variabel/kolom
housing.dtypes
```

```
[5]: CRIM       float64
     ZN         float64
     INDUS      float64
     CHAS         int64
     NOX        float64
     RM         float64
     AGE        float64
     DIS        float64
     RAD          int64
     TAX        float64
     PTRATIO    float64
     B          float64
     LISTAT     float64
     MEDV       float64
     dtype: object
```

- Pada poin 1.2 diketahui bahwa data variabel CHAS dan RAD bertipe kategori, namun dataset tersebut masih bertipe integer. Sehingga kita perlu mengubah jenis data tersebut menjadi kategorik.

```
[6]: # mengubah jenis data integer ke kategorikal pada variabel CHAS
housing["CHAS"] = housing["CHAS"].astype('category')
housing.dtypes
```

```
[6]: CRIM       float64
     ZN         float64
     INDUS      float64
     CHAS      category
     NOX        float64
     RM         float64
     AGE        float64
     DIS        float64
     RAD          int64
     TAX        float64
     PTRATIO    float64
     B          float64
     LISTAT     float64
     MEDV       float64
     dtype: object
```

```
[7]: # mengubah jenis data integer ke kategorikal pada variabel RAD
housing["RAD"] = housing["RAD"].astype('category')
housing.dtypes
```

```
[7]: CRIM       float64
     ZN         float64
     INDUS      float64
     CHAS      category
     NOX        float64
     RM         float64
     AGE        float64
     DIS        float64
     RAD       category
     TAX        float64
     PTRATIO    float64
     B          float64
     LISTAT     float64
     MEDV       float64
     dtype: object
```

- Melihat deskripsi data secara statistik dengan menggunakan command describe(). Dihasilkan output count, mean, std, min, 25%, 50%, 75%, max untuk setiap variabel/kolom.

```
[8]: # melihat deskripsi data secara statistik
housing.describe()
```

[8]:

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	TAX	PTRATIO	B	LISTAT	MEDV
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.554695	6.284634	68.574901	3.795043	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.115878	0.702617	28.148861	2.105710	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.385000	3.561000	2.900000	1.129600	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.449000	5.885500	45.025000	2.100175	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.538000	6.208500	77.500000	3.207450	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.624000	6.623500	94.075000	5.188425	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	0.871000	8.780000	100.000000	12.126500	711.000000	22.000000	396.900000	37.970000	50.000000

- Melihat seberapa banyak nilai 1 dan 0 pada variabel CHAS dengan menggunakan command value_counts(). Didapatkan output yaitu terdapat 471 nilai 0 dan 35 nilai 1, artinya terdapat banyak rumah yang salurannya tidak membatasi sungai (0).

```
[9]: # melihat seberapa banyak nilai 1 dan 0 pada variabel CHAS
housing['CHAS'].value_counts()
```

```
[9]: 0    471
     1     35
     Name: CHAS, dtype: int64
```

- Memeriksa missing value setiap variabel dengan menggunakan command isnull().any(). Missing value akan ditandakan dengan Boolean True. Didapatkan bahwa dataset The Boston Housing tidak terdapat missing value.

```
[10]: # memeriksa missing value setiap variabel
      # missing value ditandakan dengan boolean True
      # didapatkan bahwa dataset housing tidak terdapat missing value
      housing.isnull().any()
```

```
[10]: CRIM       False
      ZN         False
      INDUS      False
      CHAS       False
      NOX        False
      RM         False
      AGE        False
      DIS        False
      RAD        False
      TAX        False
      PTRATIO    False
      B          False
      LISTAT     False
      MEDV       False
      dtype: bool
```

- Melakukan transformasi data (variabel prediktor) dengan metode MinMaxScaler yang hasilnya akan disimpan pada variabel transformasi_min_max.

```
[11]: # melakukan transformasi data (variabel prediktor) metode Min-Max
      from sklearn import preprocessing
      transformasi_min_max = preprocessing.MinMaxScaler()
```

3.3. Melakukan pemodelan data dengan regresi

- Membagi dataFrame menjadi variabel target (Y) dan atribut (X). Variabel target yaitu MEDV dan sisanya merupakan atribut.

```
[12]: # membagi dataFrame menjadi variabel target (Y) dan atribut (X)
      target = 'MEDV'
      x = housing.drop('MEDV', axis=1)
      y = housing[target]
      nama_var_x = ['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','B','LSTAT']
      x = pd.DataFrame(data=transformasi_min_max.fit_transform(x), columns=nama_var_x)
```

- Membagi data menjadi data training 70% dan data testing 30% dengan mengimpor train_tes_split dari library sklearn di mana test_size=0.70 atau 70%.

```
[13]: # membagi data menjadi data training dan testing (training = 30% & testing = 70%)
      from sklearn.model_selection import train_test_split
      x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.70,random_state=1)
```

- Melakukan pemodelan data dengan regresi Linier. Kita perlu mengimpor LinearRegression dari library sklearn untuk membuat model linier. Selain itu, kita juga perlu mengimpor mean_squared_error dan r2_score dari library sklearn untuk mendapatkan nilai MSE serta R-Square. Variabel lin_model disiapkan sebagai model regresi linier. Selanjutnya, model regresi linier dibuat dengan parameter variabel independen yaitu x_train dan variabel dependen yaitu y_train.

```
[14]: # melakukan pemodelan data dengan regresi linier
      from sklearn.linear_model import LinearRegression
      from sklearn.metrics import mean_squared_error, r2_score

      lin_model = LinearRegression()
      lin_model.fit(x_train, y_train)
```

- Melakukan pemodelan data dengan regresi Polinomial. Kita perlu mengimpor mean_squared_error dan r2_score dari library sklearn untuk mendapatkan nilai MSE serta R-Square. Selain itu, kita juga perlu mengimpor PolynomialFeatures dari library sklearn untuk membuat model polinomial. Variabel polynomial_features digunakan sebagai transformasi matriks x menjadi matriks x pangkat 2, pangkat 3 hingga pangkat n. Sehingga kita akan memiliki beberapa tambahan variabel independen sebanyak n. Parameter default untuk PolynomialFeatures adalah degrees=2, jika kita membuat

degress=4 maka x_poly akan menampilkan kolom untuk nilai x_train pangkat 4 dan x_poly_test akan menampilkan kolom untuk x_test pangkat 4. Variabel x_poly digunakan sebagai hasil fit_transform (proses fit dan transform dilakukan sekaligus) dari variabel x_train. Variabel poly_model disiapkan sebagai model regresi polinomial. Selanjutnya, model regresi polinomial dibuat dengan parameter variabel independen yaitu x_poly dan variabel dependen yaitu y_train. Variabel x_poly_test digunakan sebagai hasil fit_transform dari variabel x_test.

```
[15]: # melakukan pemodelan data dengan regresi polinomial
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures

polynomial_features = PolynomialFeatures(degree=2)
x_poly = polynomial_features.fit_transform(x_train)
poly_model = LinearRegression()
poly_model.fit(x_poly, y_train)
x_poly_test = polynomial_features.fit_transform(x_test)
```

- Melakukan pemodelan data dengan regresi Ridge. Kita perlu mengimpor Ridge dari library sklearn untuk membuat model Ridge. Variabel model_ridge disiapkan sebagai model regresi Ridge dengan alpha=1. Kemudian, model regresi Ridge dibuat dengan parameter variabel independen yaitu x_train dan variabel dependen yaitu y_train.

```
[16]: # melakukan pemodelan data dengan regresi ridge
from sklearn.linear_model import Ridge

# model ridge dengan alpha=1
model_ridge = Ridge(alpha=1)
model_ridge.fit(x_train, y_train)
```

```
[16]: Ridge(alpha=1)
```

- Melakukan pemodelan data dengan regresi Lasso. Kita perlu mengimpor Lasso dari library sklearn untuk membuat model Lasso. Variabel model_lasso disiapkan sebagai model regresi Lasso dengan alpha=1. Kemudian, model regresi Lasso dibuat dengan parameter variabel independen yaitu x_train dan variabel dependen yaitu y_train.

```
[17]: # melakukan pemodelan data dengan regresi lasso
from sklearn.linear_model import Lasso

# model lasso dengan alpha=1
model_lasso = Lasso(alpha=1)
model_lasso.fit(x_train, y_train)
```

```
[17]: Lasso(alpha=1)
```

3.4. Melakukan evaluasi data training dan data testing dengan regresi

A. Regresi Linier

- Data Training. Variabel y_train_predict digunakan untuk menyimpan nilai prediksi output y_train untuk nilai input x_train dengan regresi linier. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_train dan y_train_predict. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_train dan y_train_predict.

[18]:
```
# evaluasi model untuk data training dengan regresi linier
y_train_predict = lin_model.predict(x_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))
r2 = r2_score(y_train, y_train_predict)

print("Performa model untuk data training adalah")
print("--------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model untuk data training adalah
--------------------------------------
RMSE adalah 4.9807687899948645
R2 adalah 0.7254441891569579
```

- Data Testing. Variabel y_test_predict digunakan untuk menyimpan nilai prediksi output y_test untuk nilai input x_test dengan regresi linier. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_test dan y_test_predict. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_test dan y_test_predict.

[19]:
```
# evaluasi model untuk data testing dengan regresi linier
y_test_predict = lin_model.predict(x_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))
r2 = r2_score(y_test, y_test_predict)

print("Performa model untuk data testing adalah")
print("-------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model untuk data testing adalah
-------------------------------------
RMSE adalah 4.877665892510709
R2 adalah 0.7094540602303246
```

B. Regresi Polinomial

- Data Training. Variabel y_train_predict_poly digunakan untuk menyimpan nilai prediksi output y_train untuk nilai input x_poly dengan regresi polinomial. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_train dan y_train_predict_poly. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_train dan y_train_predict_poly.

[20]:
```
# evaluasi model untuk data training dengan regresi polinomial
y_train_predict_poly = poly_model.predict(x_poly)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict_poly)))
r2 = r2_score(y_train, y_train_predict_poly)

print("Performa model polinomial derajat 2 untuk data training adalah")
print("--------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model polinomial derajat 2 untuk data training adalah
--------------------------------------
RMSE adalah 2.0488770635758873
R2 adalah 0.9535410193078007
```

- Data Testing. Variabel y_test_predict_poly digunakan untuk menyimpan nilai prediksi output y_test untuk nilai input x_poly_test dengan regresi polinomial. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_test dan y_test_predict_poly. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_test dan y_test_predict_poly.

```
[21]: # evaluasi model untuk data testing dengan regresi polinomial
y_test_predict_poly = poly_model.predict(x_poly_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict_poly)))
r2 = r2_score(y_test, y_test_predict_poly)

print("Performa model polinomial derajat 2 untuk data testing adalah")
print("--------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model polinomial derajat 2 untuk data testing adalah
--------------------------------------
RMSE adalah 12.418144675292542
R2 adalah -0.8832333796788496
```

C. Regresi Ridge

- Data Training. Variabel y_train_predict_ridge digunakan untuk menyimpan nilai prediksi output y_train untuk nilai input x_train dengan regresi ridge. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_train dan y_train_predict_ridge. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_train dan y_train_predict_ridge.

```
[22]: # evaluasi model untuk data training dengan regresi ridge
y_train_predict_ridge = model_ridge.predict(x_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict_ridge)))
r2 = r2_score(y_train, y_train_predict_ridge)

print("Performa model ridge untuk data training adalah")
print("--------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model ridge untuk data training adalah
--------------------------------------
RMSE adalah 5.15705062715862
R2 adalah 0.705665842038425
```

- Data Testing. Variabel y_test_predict_ridge digunakan untuk menyimpan nilai prediksi output y_test untuk nilai input x_test dengan regresi ridge. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_test dan y_test_predict_ridge. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_test dan y_test_predict_ridge.

```
[23]: # evaluasi model untuk data testing dengan regresi ridge
y_test_predict_ridge = model_ridge.predict(x_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict_ridge)))
r2 = r2_score(y_test, y_test_predict_ridge)

print("Performa model ridge untuk data testing adalah")
print("--------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model ridge untuk data testing adalah
--------------------------------------
RMSE adalah 4.7808135250163595
R2 adalah 0.7208778371246569
```

D. Regresi Lasso

- Data Training. Variabel y_train_predict_lasso digunakan untuk menyimpan nilai prediksi output y_train untuk nilai input x_train dengan regresi lasso. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_train dan y_train _predict_lasso. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_train dan y_train_predict_lasso.

```
[24]: # evaluasi model untuk data training dengan regresi lasso
y_train_predict_lasso = model_lasso.predict(x_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict_lasso)))
r2 = r2_score(y_train, y_train_predict_lasso)

print("Performa model lasso untuk data training adalah")
print("--------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model lasso untuk data training adalah
--------------------------------------
RMSE adalah 8.14041204183743
R2 adalah 0.26661731548896384
```

- Data Testing. Variabel y_test_predict_lasso digunakan untuk menyimpan nilai prediksi output y_test untuk nilai input x_test dengan regresi lasso. Variabel rmse digunakan untuk menyimpan nilai hitungan MSE antara y_test dan y_test_predict_lasso. Variabel r2 digunakan untuk menyimpan nilai R-Square antara y_test dan y_test_predict_lasso.

```
[25]: # evaluasi model untuk data tetsting dengan regresi lasso
y_test_predict_lasso = model_lasso.predict(x_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict_lasso)))
r2 = r2_score(y_test, y_test_predict_lasso)

print("Performa model lasso untuk data testing adalah")
print("-------------------------------------")
print('RMSE adalah {}'.format(rmse))
print('R2 adalah {}'.format(r2))
```

```
Performa model lasso untuk data testing adalah
-------------------------------------
RMSE adalah 7.707774930476879
R2 adalah 0.27448104703573784
```

3.5. Membandingkan performa model masing-masing algoritma

- Data Training

Algoritma	R-Square	RMSE
Regresi Linier	4.980768	0.725444
Regresi Polinomial	2.048877	0.953541
Regresi Ridge	5.157050	0.705665
Regresi Lasso	8.140412	0.266617

- Data Testing

Algoritma	R-Square	RMSE
Regresi Linier	4.877665	0.709454
Regresi Polinomial	12.418144	-0.883233
Regresi Ridge	4.780813	0.720877
Regresi Lasso	7.707774	0.274481

BAB 4: KESIMPULAN

RMSE digunakan untuk mengukur jarak rata-rata antara nilai prediksi dengan nilai sebenarnya. Sedangkan R-Square digunakan untuk mengukur proporsi varians dalam variabel dependen (y) yang dijelaskan oleh variabel independen (x) dalam model. Umumnya, nilai RMSE yang lebih rendah dan nilai R-Square yang lebih tinggi menunjukkan performa model yang lebih baik. Dari tabel data training pada poin 3.5 terlihat bahwa nilai R-Square lebih tinggi dibandingkan nilai RMSE yang berarti performa model data training sudah baik. Kemudian, dari tabel data testing pada poin 3.5 juga terlihat bahwa nilai R-Square lebih tinggi dibandingkan nilai RMSE yang berarti performa model data testing sudah baik.

DAFTAR PUSTAKA

algoritma. (18, February 18). *Tipe Analisis Regresi*. Diambil kembali dari algoritma learn data science by building: https://algorit.ma/blog/tipe-analisis-regresi-2022/

Algoritma. (2022, April 21). *Analisis Regresi Linier pada Machine Learning*. Retrieved from algoritma learn data science by building: https://algorit.ma/blog/data-science/analisis-regresi-linear-2022/

Delyani, G. (2021, April 28). *Kenali Analisis Regresi Linear, Salah Satu Metode Pengolahan Data yang Sering Digunakan!* Diambil kembali dari DQLab: https://dqlab.id/kenali-analisis-regresi-linear-metode-pengolahan-data-yang-sering-digunakan

Maulid, R. (2022, July 22). *Kenalan dengan Model Regresi Linear Bahasa R dan Aplikasinya*. Diambil kembali dari DQLab: https://dqlab.id/kenali-analisis-regresi-linear-metode-pengolahan-data-yang-sering-digunakan

R., A. Y. (2018, May 20). *Statistical Machine Learning*. Diambil kembali dari UGM Menara Ilmu Machine Learning: https://machinelearning.mipa.ugm.ac.id/2018/05/20/statistical-machine-learning/

Robbani, M., Agustiani, F., & Herrhyanto, N. (2019). Regresi Least Abosolute Shringkage and Selection Operator (LASSO) pada Kasus Infalsi di Indonesia Tahun 2014-2017. *Jurnal EurekaMatika*.

Setyabudi, B. D. (2017, October 23). *Kajian Analisis Regresi Polinomial Derajat Dua Berkelompok*. Diambil kembali dari ub.ac.id: http://repository.ub.ac.id/id/eprint/4129/

Stat, J. (2020, July). *Persamaan Model Polinomial*. Diambil kembali dari jagostat: https://jagostat.com/analisis-regresi/persamaan-model-regresi-polinomial

Wasilaine, T. L., Talakua, M. W., & Lesnussa, Y. A. (2014). *Model Regresi Ridge Untuk Mengatasi Model Regresi Linier Berganda Yang Mengandung Multikolinieritas*. Diambil kembali dari neliti: https://www.neliti.com/id/publications/277537/model-regresi-ridge-untuk-mengatasi-model-regresi-linier-berganda-yang-mengandun

Wasilaine, T. L., Talakua, M. W., & Lesnussa, Y. A. (2014). Model Regresi Ridge Untuk Mengatasi Model Regresi Linier Berganda Yang Mengandung Multikolinieritas. *Jurnal Barekeng*, 31-37.

Widiharih, T. (2001). *Pendekatan Regresi Polinomial Orthogonal Pada Rancangan Dua Faktor (Dengan Aplikasi Sas Dan Minitab)*. Diambil kembali dari neliti: https://www.neliti.com/publications/117934/pendekatan-regresi-polinomial-orthogonal-pada-rancangan-dua-faktor-dengan-aplika