LAPORAN TUGAS KOMPUTASI LUNAK

LOGISTIC REGRESSION

Disusun Oleh:

Rizky Akhmad Fahreza

24060121130081

Dosen Pengampu:

Khadijah, S.Kom., M.Cs.

NIP. 198903032015042002

DEPARTEMEN INFORMATIKA

FAKULTAS SAINS DAN MATEMATIKA

UNIVERSITAS DIPONEGORO

SEMARANG

2024

Perhitungan Manual

Salah satu baris data dari dataset yang sudah di normalisasi untuk dilakukan perhitungan adalah sebagai berikut:

A_id	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	0.234669	0.358370	0.922484	0.368781	0.585819	0.472770	0.452225	1

Keterangan:

- A_id: Identifikasi unik untuk setiap buah
- Size: Ukuran buah
- Weight: Berat buah
- Sweetness: Tingkat kemanisan buah
- Crunchiness: Tekstur yang menunjukkan kerenyahan buah
- Juiciness: Tingkat keberairan buah
- Ripeness: Tahap kematangan buah
- Acidity: Tingkat keasaman buah
- Quality: Kualitas keseluruhan buah (0: bad, 1: good)

Langkah-langkah:

1. Inisialisasi bobot dan bias

   Bobot awal $\omega_0, \omega_1, \ldots, \omega_6$ dan bias b yang dipilih secara acak. Untuk contoh ini:

   $\omega = [0.5, -0.2, 0.3, 0.1, -0.4, 0.2, -0.3]$ dan $b = 0.1$
2. Fungsi Linear (z)
   a. Rumus fungsi Logistic Regression untuk menghitung probabilitas di dataset ini adalah:

      $$z = \omega_0 . \text{Size} + \omega_1 . \text{Weight} + \omega_2 . \text{Sweetness} + \omega_3 . \text{Crunchiness} + \omega_4 . \text{Juiciness} + \omega_5 . \text{Ripeness} + \omega_6 . \text{Acidity} + b$$

   b. Menggunakan bobot dan data yang diberikan:

      $$z = (0.5 . 0.234669) + (-0.2 . 0.358370) + (0.3 . 0.922484) + (0.1 . 0.368781) + (-0.4 . 0.585819) + (0.2 . 0.472770) + (-0.3 . 0.452225) + 0.1$$

      $$z = 0.1173345 + (-0.071674) + 0.2767452 + 0.0368781 + (-0.2343276) + 0.094554 + (-0.1356675) + 0.1$$

      $$z = 1838427$$

3. Fungsi Sigmoid
   a. Fungsi sigmoid digunakan untuk mengubah nilai z menjadi probabilitas dengan rumus:

$$\rho = \frac{1}{1 + e^{-z}}$$

   b. Menggunakan bobot dan data yang dimiliki:

$$\rho = \frac{1}{1 + e^{-0.1848427}} = \frac{1}{1 + 0.83118} \approx 0.54609$$

4. Prediksi

   Jika probabilitas $\rho \geq 0.5$, maka prediksi kelas adalah 1 (good) dan jika $\rho < 0.5$, maka prediksi kelas adalah 0 (bad).

   Prediksi kualitas apel untuk baris data tersebut sesuai dengan hasil fungsi sigmoid adalah 1 yaitu good.

5. Fungsi Loss (Log-Loss)

   Fungsi log-loss untuk menghitung kesalahan prediksi adalah:

$$L = -(y \cdot \log(\rho) + (1 - y) \cdot \log(1-\rho))$$

   Dimana y adalah label actual dan ρ adalah probabilitas prediksi. Dengan y = 1 dan ρ = 0.54609:

$$L = -(1 \cdot \log(0.54609) + (1 - 1) \cdot \log(1-0.54609))$$

$$L = -log(0.54609) \approx 0.604$$

   Jadi nilai loss untuk satu iterasi adalah sekita 0.604.

Ringkasan :

- Probabilitas prediksi untuk baris data ini adalah $\rho \approx 0.546$
- Prediksi kelas untuk baris data ini adalah 1 (good), karena $\rho \geq 0.5$
- Loss (log-loss) untuk satu iterasi ini adalah sekitar 0.604

tugas-logistic-regression

September 18, 2024

1 Tugas Logistic Regression - Komputasi Lunak A

Rizky Akhmad Fahreza - 24060121130081

1.1 Step 1: Import semua modul/library yang dibutuhkan

[18]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, log_loss
```

Kode di atas digunakan untuk memanggil modul/library yang dibutuhkan seperti pandas dan scikit-learn.

1.2 Step 2: Load Dataset

[19]:
```
# Load Dataset
df = pd.read_csv('apple_quality.csv')

# preview dataset
df.head()
```

[19]:

	A_id	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	0	-3.970049	-2.512336	5.346330	-1.012009	1.844900	0.329840	-0.491590	good
1	1	-1.195217	-2.839257	3.664059	1.588232	0.853286	0.867530	-0.722809	good
2	2	-0.292024	-1.351282	-1.738429	-0.342616	2.838636	-0.038033	2.621636	bad
3	3	-0.657196	-2.271627	1.324874	-0.097875	3.637970	-3.413761	0.790723	good
4	4	1.364217	-1.296612	-0.384658	-0.553006	3.030874	-1.303849	0.501984	good

Kode di atas digunakan untuk membaca dataset yang berformat .csv dan menampilkan 5 baris utama dalam dataset.

[20]:
```
df.describe()
```

[20]:
```
              A_id         Size       Weight    Sweetness  Crunchiness  \
count  4000.000000  4000.000000  4000.000000  4000.000000  4000.000000
mean   1999.500000    -0.503015    -0.989547    -0.470479     0.985478
std    1154.844867     1.928059     1.602507     1.943441     1.402757
min       0.000000    -7.151703    -7.149848    -6.894485    -6.055058
25%     999.750000    -1.816765    -2.011770    -1.738425     0.062764
50%    1999.500000    -0.513703    -0.984736    -0.504758     0.998249
75%    2999.250000     0.805526     0.030976     0.801922     1.894234
max    3999.000000     6.406367     5.790714     6.374916     7.619852

         Juiciness     Ripeness      Acidity
count  4000.000000  4000.000000  4000.000000
mean      0.512118     0.498277     0.076877
std       1.930286     1.874427     2.110270
min      -5.961897    -5.864599    -7.010538
25%      -0.801286    -0.771677    -1.377424
50%       0.534219     0.503445     0.022609
75%       1.835976     1.766212     1.510493
max       7.364403     7.237837     7.404736
```

Kode di atas digunakan untuk melihat informasi statistik terkait dataset yang digunakan

1.3 Step 3: Data Preparation

1.3.1 1) Mengubah label string menjadi numerik

[21]:
```
# Inisialisasi LabelEncoder
label_encoder = LabelEncoder()

# Aplikasikan LabelEncoder ke kolom label
df['Quality'] = label_encoder.fit_transform(df['Quality'])
df.head()
```

[21]:
```
   A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
0     0 -3.970049 -2.512336   5.346330    -1.012009   1.844900  0.329840
1     1 -1.195217 -2.839257   3.664059     1.588232   0.853286  0.867530
2     2 -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -0.038033
3     3 -0.657196 -2.271627   1.324874    -0.097875   3.637970 -3.413761
4     4  1.364217 -1.296612  -0.384658    -0.553006   3.030874 -1.303849

    Acidity  Quality
0 -0.491590        1
1 -0.722809        1
```

```
2  2.621636        0
3  0.790723        1
4  0.501984        1
```

Kode di atas digunakan untuk memanggil function LabelEncoder yang akan digunakan untuk mengubah seluruh data pada kolom Quality yang tadinya bertipe string menjadi numerik.

1.3.2 2) Menormalisasi data

[22]:
```
from sklearn.preprocessing import MinMaxScaler

# Inisialisasi MinMaxScaler
scaler = MinMaxScaler()

# Mengaplikasikan MinMaxScaler ke kolom selain ID dan label
df_scaled = df.copy()
df_scaled[df.columns[1:-1]] = scaler.fit_transform(df[df.columns[1:-1]])

# Menampilkan data
df_scaled.head()
```

[22]:
```
   A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
0     0  0.234669  0.358370   0.922484     0.368781   0.585819  0.472770
1     1  0.439331  0.333107   0.795706     0.558928   0.511408  0.513807
2     2  0.505948  0.448092   0.388567     0.417732   0.660388  0.444693
3     3  0.479014  0.376971   0.619422     0.435629   0.720370  0.187052
4     4  0.628107  0.452317   0.490589     0.402347   0.674814  0.348084

    Acidity  Quality
0  0.452225        1
1  0.436185        1
2  0.668192        0
3  0.541180        1
4  0.521150        1
```

Kode di atas digunakan untuk memanggil function MinMaxScaler yang akan digunakan untuk menormalisasi data yang terdapat di dataset.

1.4 Step 4: Splitting Data

[23]:
```
# memisahkan features (X) and target (y)
X = df_scaled.drop(['A_id', 'Quality'], axis=1)
y = df_scaled['Quality']

# Membagi dataset (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=None)
```

Kode di atas digunakan untuk memisahkan features dataset untuk X dan target y. Setelahnya, dataset dibagi dengan persentase 80% training dan 20% testing.

[24]:
```
X_train
```

[24]:

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity
2495	0.503341	0.398464	0.597717	0.610016	0.525783	0.573745	0.440696
236	0.354650	0.504698	0.485345	0.530365	0.530073	0.665402	0.533187
2042	0.279632	0.517434	0.593826	0.314068	0.454417	0.596742	0.367199
3718	0.727149	0.737092	0.327921	0.683318	0.246419	0.495631	0.584907
3820	0.565965	0.430831	0.432798	0.359495	0.654391	0.327503	0.694611
…	…	…	…	…	…	…	…
2603	0.568449	0.466420	0.431272	0.369769	0.857433	0.354006	0.670364
2428	0.701647	0.216945	0.511903	0.501995	0.214772	0.503981	0.280920
3103	0.319300	0.581520	0.499231	0.663179	0.282367	0.669364	0.415790
872	0.519589	0.639013	0.538500	0.436133	0.446548	0.291489	0.405038
830	0.297634	0.450149	0.621417	0.566461	0.648130	0.411635	0.348221

[3200 rows x 7 columns]

[25]:
```
X_test
```

[25]:

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness
157	0.389139	0.467157	0.527636	0.648152	0.282390	0.586473
2578	0.524435	0.412120	0.503718	0.520060	0.430612	0.666146
439	0.427663	0.559930	0.344001	0.627004	0.209334	0.414145
3884	0.396463	0.366222	0.540572	0.542799	0.424938	0.576626
2937	0.684124	0.401138	0.165399	0.447126	0.563722	0.658175
…	…	…	…	…	…	…
3861	0.329902	0.594795	0.566719	0.639748	0.289463	0.461367
285	0.494920	0.381941	0.318466	0.499724	0.461600	0.703624
3521	0.275088	0.620977	0.599878	0.511910	0.630128	0.367980
3048	0.380646	0.716721	0.448773	0.566384	0.605000	0.418765
1384	0.499204	0.535888	0.561272	0.483746	0.382093	0.571848

```
        Acidity
157   0.493251
2578  0.504715
439   0.485692
3884  0.545733
2937  0.336878
…          …
3861  0.552741
285   0.514738
3521  0.445356
3048  0.355302
1384  0.436316

[800 rows x 7 columns]
```

[26]:
```
y_train
```

[26]:
```
2495    1
236     0
2042    0
3718    1
3820    0
       ..
2603    0
2428    0
3103    0
872     1
830     1
Name: Quality, Length: 3200, dtype: int64
```

[27]:
```
y_test
```

[27]:
```
157     0
2578    1
439     0
3884    0
2937    0
       ..
3861    0
285     0
3521    1
3048    1
1384    1
Name: Quality, Length: 800, dtype: int64
```

1.5 Step 5: Melakuakan Training Model Logistic Regression

[28]:
```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

[28]:
```
LogisticRegression(max_iter=1000)
```

Kode di atas digunakan untuk memanggil model LogisticRegression dan melakukan training dengan maksimal 1000 iterasi.

1.6 Step 6: Melakukan prediksi

[29]:
```
y_pred = model.predict(X_test)
```

[30]:
```
# Menghitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 74.88%
```

Kode di atas digunakan untuk melakukan prediksi dan melihat skor accuracynya.

1.7 Step 7: Menghitung Log-Loss

[31]:
```
# Predict on the test set (probability predictions)
y_pred_probs = model.predict_proba(X_test)[:, 1]  # Predicted probabilities for␣
↪class 1
```

[32]:
```
# Menghitung Log-Loss
logloss = log_loss(y_test, y_pred_probs)
print(f'Log-Loss: {logloss:.4f}')
```

```
Log-Loss: 0.5256
```

Kode di atas digunakan untuk menghitung log-loss dengan tahap awal menghitung probabilitas banyaknya data apel yang memiliki kualitas baik terhadap seluruh apel. Kemudian dilakukan perhitungan log-loss dengan menggunakan function log-loss.

1.8 Step 8: Menampilkan Confusion Matrix

[33]:
```
# Generate confusion matrix
confusion_matrix(y_test, y_pred)
```

[33]:
```
array([[296, 112],
       [ 89, 303]])
```

Kode di atas digunakan untuk menampilkan confussion matriks dari hasil prediksi model.