



Binary Classification using Logistic Regression on Determining Risk of Heart Disease

Machine Learning/LD01

Dr. Ir. Diaz D. Santika, M. Sc.

Diusulkan oleh :


Mikhael Adiputra	2301957572	Binusian 2023
Bobby Ravel Moreno	2301924933	Binusian 2023
Ryan Razaan Gunawan	2301878290	Binusian 2023

Dataset Overview

Pada project ini, kelompok kami menggunakan dataset heart.csv yang kami dapatkan dari website kaggle.com. Link : <https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset>.

Dataset ini terdiri dari 303 row data dengan masing-masing memiliki 14 kolom yang terbagi dari 13 features dan 1 target berupa binary output (0 atau 1).

Berikut metadata dari heart.csv.

Metadata		
Usage Information	License	CC0: Public Domain ⓘ
	Visibility	👁 Public
Provenance	Sources	Online
	Collection methodology	Crawling
Maintainers	Dataset owner	 Rashik Rahman
Updates	Expected update frequency	Annually
	Last updated	2021-03-22
	Date created	2021-03-22
	Current version	Version 2

Code Analysis

```
[1]: # Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.optimize as opt
%matplotlib inline
```

Library yang diimport pertama adalah:

- Numpy untuk perhitungan operasi
- Pandas untuk olah dataset
- Pyplot untuk visualisasi data
- %matplotlib inline adalah fungsi untuk memudahkan visualisasi data pada file notebook

```
[2]: dataset = pd.read_csv('heart.csv', sep = ",")
dataset.head(10)
```

```
[2]:
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

Dataset dalam bentuk csv kami akses menggunakan fungsi `read_csv` dari pandas. Kami menambahkan `sep = ","` disamping nama file karena file ini diakses pada Mac Device yang mana separatornya berbeda dari windows.

```
[3]: dataset.describe()
```

```
[3]:
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

Berikut adalah visualisasi data dengan panggilan `.describe()` dari library pandas.

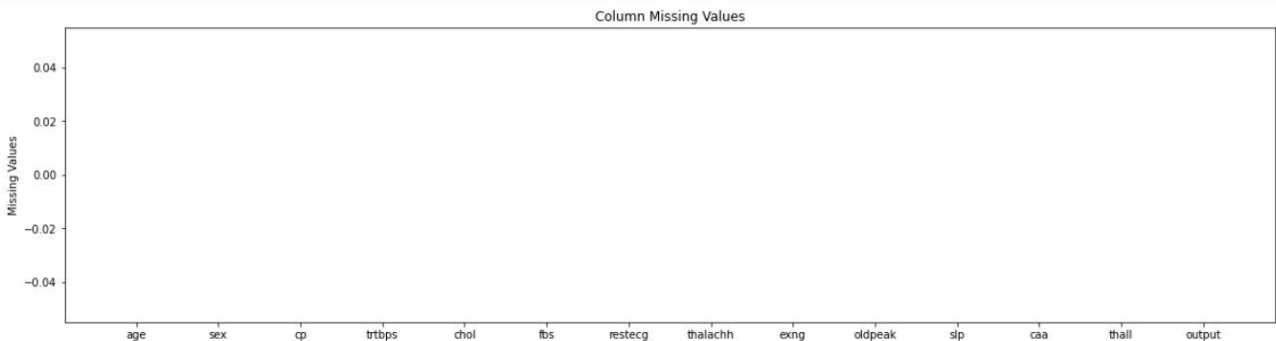
```
[4]: # Checking for missing values
def visualize_null(dataset):
    column_name = list(dataset.keys())
    values = list(dataset.isnull().sum())

    fig = plt.figure(figsize = (20, 5))

    # creating the bar plot
    plt.bar(column_name, values, color = 'maroon', width = 0.4)

    plt.ylabel("Missing Values")
    plt.title("Column Missing Values")
    plt.show()
```

```
[5]: visualize_null(dataset)
```



Untuk mengecek keberadaan Null Value, kami melakukan visualisasi terhadap isi dari dataset menggunakan fungsi isnull() yang menunjukkan Null Value dalam data.

Karena tidak ditemukan Null dalam dataset maka kami bisa melewati proses imputation.

```
[6]: correlations = dataset.corr()

print(correlations)
```

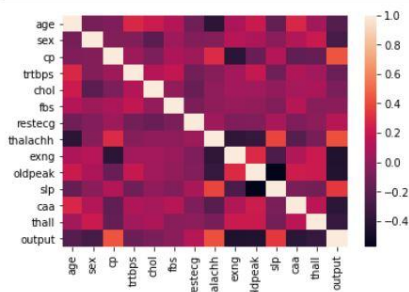
	age	sex	cp	trtbps	chol	fbs
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444
trtbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189
thalachh	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567
exng	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747
slp	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894
caa	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979
thall	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019
output	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046

	restecg	thalachh	exng	oldpeak	slp	caa
age	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326
sex	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261
cp	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053
trtbps	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389
chol	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511
fbs	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979
restecg	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042
thalachh	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177
exng	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739
oldpeak	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682
slp	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155
caa	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000
thall	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832
output	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724

	thall	output
age	0.068001	-0.225439
sex	0.210041	-0.280937
cp	-0.161736	0.433798
trtbps	0.062210	-0.144931
chol	0.098803	-0.085239
fbs	-0.032019	-0.028046
restecg	-0.011981	0.137230
thalachh	-0.096439	0.421741
exng	0.206754	-0.436757
oldpeak	0.210244	-0.430696
slp	-0.104764	0.345877
caa	0.151832	-0.391724
thall	1.000000	-0.344029
output	-0.344029	1.000000

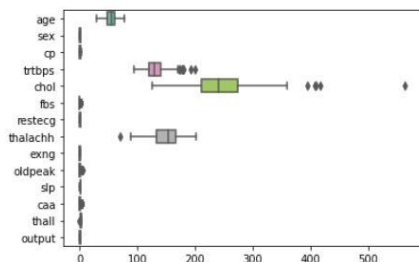
Untuk menghitung korelasi dari masing-masing kolom data, kami memanggil fungsi `.corr()` dari pandas dan berikut adalah hasilnya.

```
[7]: import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(correlations)
plt.show()
```



Untuk memudahkan menganalisis nilai kolerasi, kami visualisasikan nilai korelasi ke dalam bentuk graph dengan tool heatmap dari seaborn.

```
[8]: # Checking for outliers
import seaborn as sns
outliers = sns.boxplot(data=dataset, orient="h", palette="Set2")
```



Setelah itu, kami juga memvisualisasikan outlier dengan bantuan `seaborn.boxplot` dengan orientasi horizontal yang dapat dilihat pada gambar diatas.

```
[9]: # Removing outliers using IQR
Q1 = dataset.quantile(0.25)
Q3 = dataset.quantile(0.75)
IQR = Q3-Q1
print(IQR)

age      13.5
sex       1.0
cp        2.0
trtbps   20.0
chol     63.5
fbs       0.0
restecg   1.0
thalachh  32.5
exng      1.0
oldpeak   1.6
slp       1.0
caa       1.0
thall     1.0
output    1.0
dtype: float64
```

Dalam menghilangkan outlier pada data, kami menggunakan metode Interquartile Range (IQR). Dimana Q1 menandakan batas bawah dan Q3 menandakan batas atas dari keseluruhan data point dalam suatu kolom. Seluruh data diluar range tersebut dianggap sebagai outlier.

```
[10]: # Remove Outlier from the dataset
dataset = dataset[~((dataset < (Q1 - 1.5 * IQR)) | (dataset > (Q3 + 1.5 * IQR))).any(axis=1)]
dataset.shape

[10]: (228, 14)
```

Di atas merupakan rumus umum dalam menghilangkan outlier dengan metode IQR. Dimana seluruh data yang berada di bawah Q1 dan di atas Q3 akan dihilangkan.

```
[11]: dataset.describe()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
count	228.000000	228.000000	228.000000	228.000000	228.000000	228.0	228.000000	228.000000	228.000000	228.000000	228.000000	228.000000	228.000000	228.000000
mean	53.333333	0.675439	0.942982	128.671053	242.372807	0.0	0.548246	151.070175	0.315789	0.946053	1.451754	0.47807	2.315789	0.578947
std	9.229016	0.469241	1.020190	15.349142	44.329827	0.0	0.516125	22.492963	0.465852	1.035422	0.587945	0.69893	0.560299	0.494814
min	29.000000	0.000000	0.000000	94.000000	131.000000	0.0	0.000000	88.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
25%	45.000000	0.000000	0.000000	120.000000	209.750000	0.0	0.000000	137.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	54.000000	1.000000	1.000000	130.000000	239.000000	0.0	1.000000	155.000000	0.000000	0.600000	1.500000	0.000000	2.000000	1.000000
75%	60.000000	1.000000	2.000000	140.000000	269.250000	0.0	1.000000	168.250000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	76.000000	1.000000	3.000000	170.000000	360.000000	0.0	2.000000	202.000000	1.000000	4.000000	2.000000	2.000000	3.000000	1.000000

Setelah semua outlier sudah teratasi, kami sekali lagi mengecek dataset menggunakan .describe(), dan dapat terlihat, dari total count data yang awalnya berjumlah 303, telah berkurang menjadi 228 total data yang akan kami gunakan selanjutnya.

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

keys = dataset.keys()

scaler = MinMaxScaler()
scaler.fit(dataset)
dataset = scaler.transform(dataset)
dataset = pd.DataFrame(dataset, columns = keys)
```

Normalisasi terhadap dataset dilakukan dengan MinMaxScaler dari scikit-learn. Hal ini dilakukan untuk mengurangi redundansi data dengan begitu model training akan bekerja lebih baik.

[14]:	cp	thalachh	slp	restecg	fbs	ttrbtp	chol	age	sex	output
0	0.666667	0.868421	0.0	0.5	0.0	0.473684	0.519651	0.170213	1.0	1.0
1	0.333333	0.736842	1.0	0.0	0.0	0.473684	0.318777	0.255319	0.0	1.0
2	0.333333	0.789474	1.0	0.5	0.0	0.342105	0.458515	0.574468	1.0	1.0
3	0.000000	0.657895	1.0	0.5	0.0	0.342105	0.973799	0.595745	0.0	1.0
4	0.000000	0.526316	0.5	0.5	0.0	0.605263	0.266376	0.595745	1.0	1.0

```
[15]: # Split dataset 80% for training and 20% for testing

train_dataset = finalized_dataset.sample(frac=0.8, random_state=1)
test_dataset = finalized_dataset.drop(train_dataset.index)


# Prepare features and labels for train and test dataset
train_x = train_dataset.loc[:, train_dataset.columns != "output"]
test_x = test_dataset.loc[:, test_dataset.columns != "output"]

train_y = train_dataset['output'].values
test_y = test_dataset['output'].values
```

[16]:	cp	thalachh	slp	restecg	fbs	trtbps	chol	age	sex
39	0.666667	0.596491	1.0	0.0	0.0	0.578947	0.550218	0.382979	1.0
169	0.000000	0.456140	0.5	0.5	0.0	0.315789	0.384279	0.212766	1.0
93	0.000000	0.824561	1.0	0.0	0.0	0.578947	0.611354	0.638298	1.0
62	0.666667	0.675439	0.5	0.5	0.0	0.368421	0.358079	0.297872	0.0
198	0.000000	0.061404	0.5	0.5	0.0	0.381579	0.659389	0.510638	1.0
114	0.666667	0.245614	0.5	1.0	0.0	0.605263	0.288210	1.000000	0.0
38	0.666667	0.807018	1.0	0.0	0.0	0.605263	0.454148	0.319149	1.0
123	0.666667	0.561404	0.5	0.0	0.0	0.684211	0.641921	0.787234	0.0
106	0.333333	0.657895	1.0	0.5	0.0	0.421053	0.764192	0.255319	0.0
89	1.000000	0.649123	0.5	0.0	0.0	0.342105	0.270742	0.574468	1.0

```
[17]: array([[1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1.,
            1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1.,
            1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 1.,
            1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 1.,
            1., 0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 1., 1.,
            1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0., 0.,
            1., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
            1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0.,
            1., 0., 1., 0., 0., 1., 1., 1., 1., 1., 0., 1., 0., 0., 1.,
            0., 1., 0., 0., 1., 1., 1., 1., 0., 1., 0., 0., 0., 1., 0., 1.,
            0., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 0.]])
```

Fungsi aktivasi yang digunakan adalah sigmoid karena cocok untuk digunakan pada binary classification.

```
[19]: def calculate_cost(theta, X, y, r_lambda):
    m = X.shape[0]
    h = sigmoid(theta, X)
    cost = (1/m)*(-y*np.log(h)-(1-y)*np.log(1-h)).sum() + (r_lambda / (2*m))*np.square(theta).sum()
    cost -= (r_lambda / (2*m)) * theta[0]**2
    return cost
```

Fungsi di atas digunakan untuk menghitung nilai cost/loss yang merupakan perbedaan nilai antara prediksi dengan target sebenarnya.

```
[20]: def calculate_grad(theta, X, y, r_lambda):
    m = X.shape[0]
    h = sigmoid(theta, X)
    grad = np.matmul(X.transpose(), h - y)
    grad += (r_lambda/m) * theta
    grad[0] -= (r_lambda/m) * theta[0]
    return grad
```

Di atas merupakan fungsi gradient descent yang kami gunakan untuk mengupdate nilai weights (tetha) yang menghasilkan loss minimum.

```
[21]: # M = len dataset
# n = count of weight
# y = label
# X = dataset Features
def logistic_regression(theta, X, y, lr, epoch, r_lambda):
    m = X.shape[0]
    costs_function = []
    test_cost_function = []

    for i in range(epoch):
        costs_function.append(calculate_cost(theta, X, y, r_lambda))
        theta -= lr * (1/m)*calculate_grad(theta, X, y, r_lambda)
        x_graph = np.arange(0, epoch, 1);

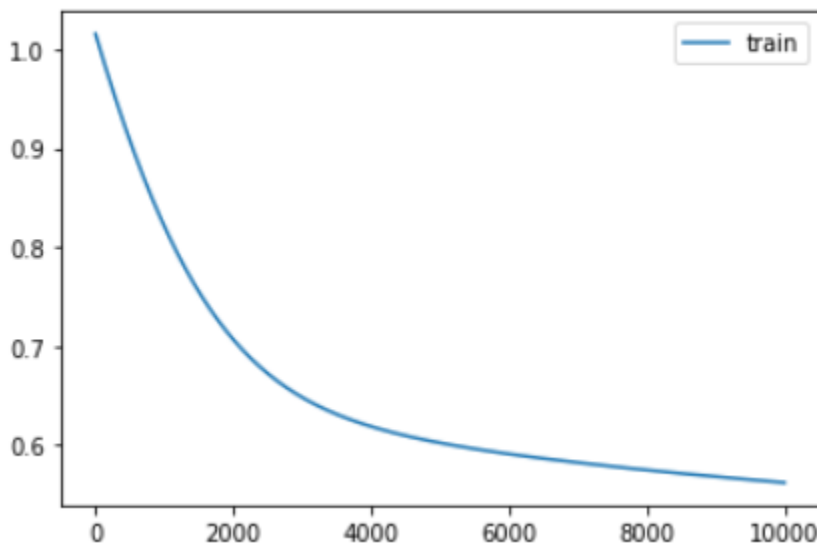
    return theta, costs_function, x_graph
```

Model training yang kami gunakan untuk binary classification adalah Logistic Regression.

```
[22]: theta = np.random.rand(train_x.shape[1])
theta, cost_function, x_graph = logistic_regression(theta, train_x.values, train_y, 0.001, 10000, 1)

plt.plot(x_graph, cost_function, label = 'train')
plt.legend()
```

[22]: <matplotlib.legend.Legend at 0x7fdccc58d450>



Theta merupakan weights yang akan digunakan untuk menyesuaikan nilai prediksi hingga sesuai target. Learning Rate kami set 0.001 dan jumlah iterasi (epoch) kami set 10000.

Grafik di atas adalah hasil visualisasi penurunan nilai loss pada saat training berjalan selama 10000 iterasi, dimana loss terakhir mencapai 0.55

```
[23]: def predict(theta, dataset, threshold):  
      prediction = sigmoid(theta, dataset)  
      y_predicted = (prediction>=threshold).astype(int)  
      return y_predicted
```

Fungsi predict digunakan untuk memprediksi nilai dari hasil dataset testing menggunakan model yang weight nya telah disesuaikan saat fase training. Penggunaan dataset yang berbeda dari fase training ini dilakukan agar model mencoba prediksi dengan data yang benar-benar baru.

```
[24]: from sklearn.metrics import accuracy_score  
  
      y_pred = predict(theta, test_x.values, 0.5)  
      accuracy = accuracy_score(test_y, y_pred)  
      print(accuracy)  
  
0.7608695652173914
```

Evaluasi performa model kami lakukan dengan bantuan fungsi accuracy score dari scikit-learn. Nilai akurasi dihitung dengan membandingkan hasil prediksi dengan hasil asli dari data testing. Model kami mendapat akurasi senilai 76%.

```
[25]: from sklearn.metrics import confusion_matrix  
      confusion_matrix(test_y, y_pred)  
  
[25]: array([[10, 10],  
            [ 1, 25]])
```

Confusion matrix pada gambar di atas, menjelaskan hasil dari testing kami. Jawaban benar kami adalah 35 prediksi dari 46 kemungkinan. True positive sejumlah 10 jawaban dan true negative sejumlah 25 jawaban.

```
[26]: from sklearn.linear_model import LogisticRegression  
  
      clf = LogisticRegression(random_state=0).fit(train_x, train_y)
```

Setelah itu, kami juga membandingkan performa model kami dengan model yang sudah tersedia, yaitu LogisticRegression dari library sklearn.

```
[27]: y_pred_ml = clf.predict(test_x)  
      accuracy_ml = accuracy_score(test_y, y_pred_ml)  
      print(accuracy_ml)  
  
0.7608695652173914
```

Dapat dilihat, akurasi dari sklearn LogisticRegression, berjumlah 76% atau sama persis dengan akurasi dari model yang kami rancang.

```
[28]: confusion_matrix(test_y, y_pred_ml)  
  
[28]: array([[12,  8],  
            [ 3, 23]])
```

Terakhir, berikut adalah visualisasi hasil training dari sklearn LogisticRegression menggunakan confusion matrix. Dapat dilihat, terdapat 12 prediksi positif benar, dengan 8 prediksi positif salah, dan 23 prediksi negatif benar, dengan 3 prediksi negatif salah. Dengan gabungan prediksi benar sebanyak 25 data dan 11 total prediksi salah.