TUGAS PERTEMUAN 2

MACHINE LEARNING

Nama : **Bima Rizki Widiatna**

NPM : **41155050210061**

Prodi/Kelas : **Teknik Informatika/A2**

1. *Simple Linear Regression* dengan *Scikit-Learn*

1.1. Sample Dataset

[7]:

```
import pandas as pd

pizza = {'diameter': [6, 8, 10, 14, 18],
         'harga': [7, 9, 13, 17.5, 18]}

pizza_df = pd.DataFrame(pizza)
pizza_df
```

[7]:

	diameter	harga
0	6	7.0
1	8	9.0
2	10	13.0
3	14	17.5
4	18	18.0

[8]:

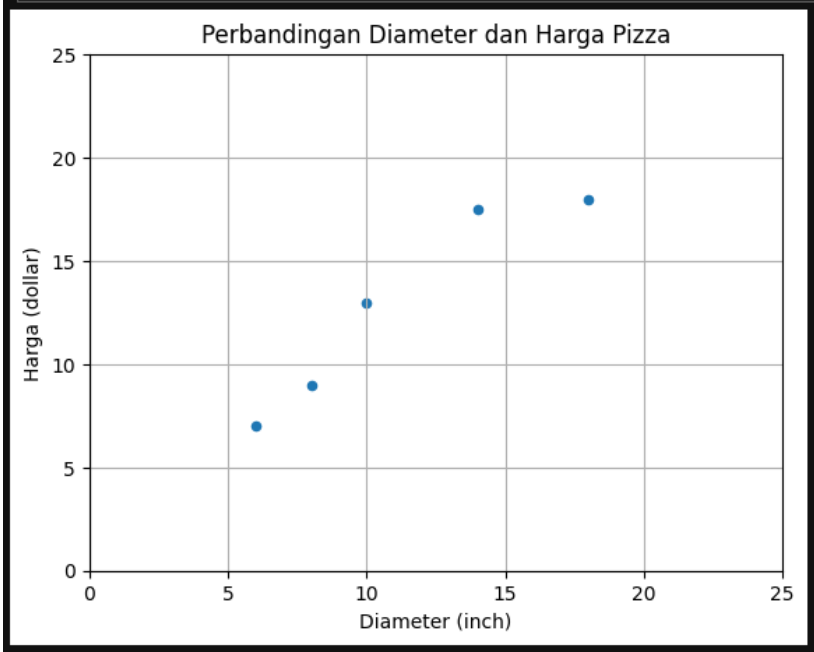
```
print('Bima Rizki')
```

```
Bima Rizki
```

1.2. Visualisasi Dataset

[14]:

```
import matplotlib.pyplot as plt

pizza_df.plot(kind='scatter', x='diameter', y='harga')

plt.title('Perbandingan Diameter dan Harga Pizza')
plt.xlabel('Diameter (inch)')
plt.ylabel('Harga (dollar)')
plt.xlim(0, 25)
plt.ylim(0, 25)
plt.grid(True)
plt.show()
print('Bima Rizki')
```




```
Bima Rizki
```

1.3. Transformasi Dataset

[16]:

```
import numpy as np

X = np.array(pizza_df['diameter'])
y = np.array(pizza_df['harga'])

print(f'X: {X}')
print(f'y: {y}')
print('Bima Rizki')
```

```
X: [ 6  8 10 14 18]
y: [ 7.   9.  13.  17.5 18. ]
Bima Rizki
```

```
[17]:
X = X.reshape(-1, 1)
X.shape

[17]:
(5, 1)

[20]:
X

[20]:
array([[ 6],
       [ 8],
       [10],
       [14],
       [18]])
```

1.4. *Training Simple Linear Regression Model*

```
[22]:
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X, y)

[22]:
LinearRegression
LinearRegression()
```

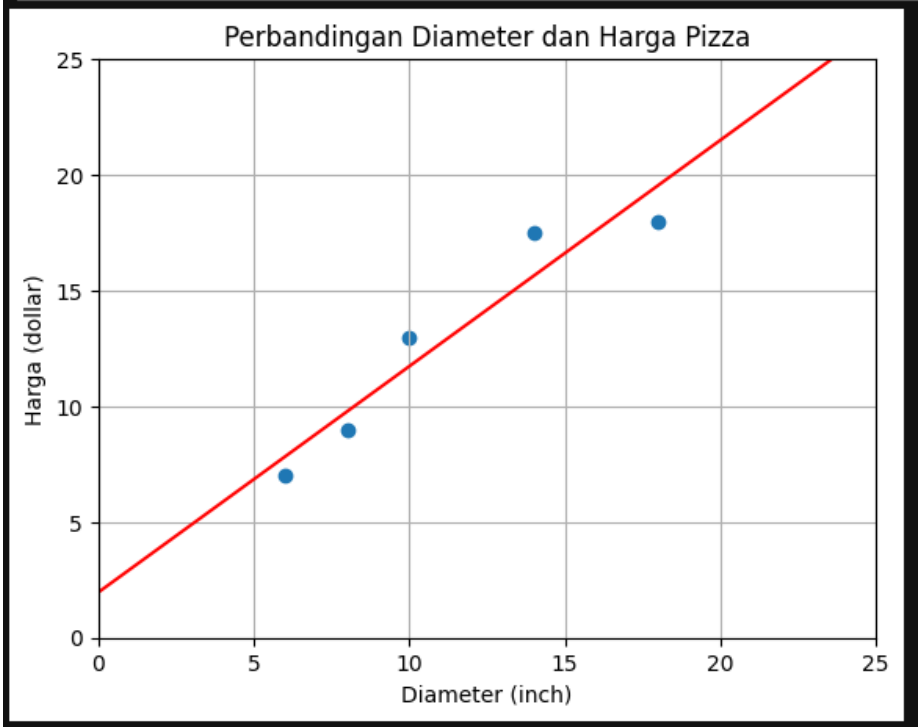
1.5. Visualisasi Simple Linear Regression Model

[23]:

```
X_vis = np.array([0, 25]).reshape(-1, 1)
y_vis = model.predict(X_vis)
```

[24]:

```
plt.scatter(X, y)
plt.plot(X_vis, y_vis, '-r')

plt.title('Perbandingan Diameter dan Harga Pizza')
plt.xlabel('Diameter (inch)')
plt.ylabel('Harga (dollar)')
plt.xlim(0, 25)
plt.ylim(0, 25)
plt.grid(True)
plt.show()

print('Bima Rizki')
```




```
Bima Rizki
```

Formula Linear Regression :

$$y = \alpha + \beta x$$

y = response variable (target)

x = explanatory variable (feature)

α = intercept

β = slope

Intercept merupakan titik pada sumbu y, dimana garis linear yang terbentuk menabrak suatu titik pada sumbu y. Nilai ***Slope*** akan berperngaruh pada tingkat kemiringan garis linear yang terbentuk, dimana nilai *slope* 0 akan menghasilkan garis horizontal.

```
[27]:
print(f'intercept: {model.intercept_}')
print(f'slope: {model.coef_}')

intercept: 1.965517241379315
slope: [0.9762931]
```

Gambar di atas adalah cara untuk menampilkan nilai *intercept* dan *slope*.

1.6. Kalkulasi Nilai *Slope*

Formula mencari nilai *slope*

$$\boldsymbol{\beta} = \frac{\boldsymbol{cov}(\boldsymbol{x}, \boldsymbol{y})}{\boldsymbol{var}(\boldsymbol{x})}$$

cov* = *covariance

var* = *variamce

```
[30]:
print(f'X:\n{X}\n')
print(f'X flatten: {X.flatten()}\n')
print(f'y: {y}')
print('\nBima Rizki')

X:
[[ 6]
 [ 8]
 [10]
 [14]
 [18]]

X flatten: [ 6  8 10 14 18]

y: [ 7.   9.  13.  17.5 18. ]

Bima Rizki
```

Pada gambar di atas array 2 dimensi di ubah ke bentuk asalnya menjadi array 1 dimensi dengan menggunakan *flatten.*

1.6.1. Variance

```
[31]:
variance_x = np.var(X.flatten(), ddof=1)

print(f'variance: {variance_x}')
print('Bima Rizki')
```

```
variance: 23.2
Bima Rizki
```

1.6.2. Covariance

```
[32]:
np.cov(X.flatten(), y)
```

```
[32]:

array([[23.2 , 22.65],
       [22.65, 24.3 ]])
```

np.cov digunakan untuk menampilkan matriks *covariance*. Nilai yang akan di ambil adalah diagonal dari matriksnya, yaitu nilai 22,65 pada gambar diatas. Gambar berikut adalah nilai dari *covariance*-nya.

```
[33]:
covariance_xy = np.cov(X.flatten(), y)[0][1]

print(f'covariance: {covariance_xy}')
print('\nBima Rizki')
```

```
covariance: 22.650000000000002

Bima Rizki
```

1.6.3. Slope

Berikut nilai *slope* dari hasil perhitungan nilai *covariance* dibagi nilai *variance* yang sebelumnya telah dihitung.

```
[34]:
slope = covariance_xy / variance_x

print(f'slope: {slope}')
print('\nBima Rizki')
```

```
slope: 0.976293103448276

Bima Rizki
```

1.7. Kalkulasi Nilai *Intercept*

Formula mencari nilai *intercept*

$$\alpha = \overline{y} - \beta\overline{x}$$

$\overline{y}$ = *nilai rata rata target*

β = *nilai slope*

$\overline{x}$ = *nilai rata rata x atau feature*

```
[35]:
intercept = np.mean(y) - slope * np.mean(X)

print(f'intercept: {intercept}')
print('\nBima Rizki')
```

```
intercept: 1.9655172413793096

Bima Rizki
```

1.8. Prediksi Harga Pizza Dengan Simple Linear Regression Model

```
[36]:
diameter_pizza = np.array([12, 20, 23]).reshape(-1, 1)
diameter_pizza
```

```
[36]:

array([[12],
       [20],
       [23]])
```

```
[37]:
prediksi_harga = model.predict(diameter_pizza)
prediksi_harga
```

```
[37]:

array([13.68103448, 21.49137931, 24.42025862])
```

```
[40]:
for dmtr, hrg in zip(diameter_pizza, prediksi_harga):
    print(f'Diameter: {dmtr} \nPrediksi Harga: {hrg}\n')
```

```
Diameter: [12]
Prediksi Harga: 13.681034482758621

Diameter: [20]
Prediksi Harga: 21.491379310344826

Diameter: [23]
Prediksi Harga: 24.42025862068965


[41]:
print('Bima Rizki')
```

```
Bima Rizki
```

1.9. Evaluasi Model Dengan Coefficient Of Determination | R Squared

```
[42]:
X_train = np.array([6, 8, 10, 14, 18]).reshape(-1, 1)
y_train = np.array([7, 9, 13, 17.5, 18])

X_test = np.array([8, 9, 11, 16, 12]).reshape(-1, 1)
y_test = np.array([11, 8.5, 15, 18, 11])
```

```
[44]:
model = LinearRegression()
model.fit(X_train, y_train)
```

```
[44]:
LinearRegression
LinearRegression()
```

```
[45]:
from sklearn.metrics import r2_score

y_pred = model.predict(X_test)

r_squared = r2_score(y_test, y_pred)

print(f'R-squared: {r_squared}')
print('\nBima Rizki')
```

```
R-squared: 0.6620052929422553

Bima Rizki
```

Semakin nilai *R-squared* mendekati 1 semakin baik, semakin menjauhi 1 atau mendekati 0 semakin buruk. Jika model yang dimiliki memiliki kualitas yang buruk sekali, nilai *R-squared* akan berisi nilai negatif.

1.10. Kalkulasi Nilai R Squared | Coefficient Of Determination

Rumus mencari nilai *R-squared:*

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$SS_{res} = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

```
[46]:
ss_res = sum([(y_i - model.predict(x_i.reshape(-1, 1))[0])**2
              for x_i, y_i in zip(X_test, y_test)])

print(f'ss_res: {ss_res}')
print('\nBima Rizki')
```

```
ss_res: 19.1980993608799

Bima Rizki
```

SS_{res} = *Sum Squared Residual*

$$\boldsymbol{SS_{tot}} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\bar{y}})^2$$

```
[47]:
mean_y = np.mean(y_test)
ss_tot = sum([(y_i - mean_y)**2 for y_i in y_test])

print(f'ss_tot: {ss_tot}')
print('\nBima Rizki')
```

```
ss_tot: 56.8

Bima Rizki
```

SS_{tot} = *Sum Squared Total*

Berikut adalah perhitungan nilai $\boldsymbol{R^2}$

```
[49]:
r_squared = 1 - (ss_res / ss_tot)
print(f'R-Squared: {r_squared}')
print('\nBima Rizki')
```

```
R-Squared: 0.6620052929422553

Bima Rizki
```

2. *Multiple Linear Regression & Polynomial Regression*

2.1. Persiapan Sample Dataset

2.1.1. Train Dataset

```
[1]:
import pandas as pd

pizza = {'diameter': [6, 8, 10, 14, 18],
         'n_topping': [2, 1, 0, 2, 0],
         'harga': [7, 9, 13, 17.5, 18]}

train_pizza_df = pd.DataFrame(pizza)
train_pizza_df
```

[1]:

	diameter	n_topping	harga
0	6	2	7.0
1	8	1	9.0
2	10	0	13.0
3	14	2	17.5
4	18	0	18.0

2.1.2. Testing Dataset

```
[3]:
pizza = {'diameter': [8, 9, 11, 16, 12],
         'n_topping': [2, 0, 2, 2, 0],
         'harga': [11, 8.5, 15, 18, 11]}

test_pizza_df = pd.DataFrame(pizza)
test_pizza_df
```

[3]:

	diameter	n_topping	harga
0	8	2	11.0
1	9	0	8.5
2	11	2	15.0
3	16	2	18.0
4	12	0	11.0

```
[4]:
print('Bima Rizki')
```

```
Bima Rizki
```

2.2. *Preprocessing* Dataset

[7]:

```
import numpy as np

X_train = np.array(train_pizza_df[['diameter', 'n_topping']])
y_train = np.array(train_pizza_df['harga'])

print(f'X_train:\n{X_train}')
print(f'y_train: {y_train}')
print('\nBima Rizki')
```

```
X_train:
[[ 6  2]
 [ 8  1]
 [10  0]
 [14  2]
 [18  0]]
y_train: [ 7.   9.  13.  17.5 18. ]

Bima Rizki
```

[10]:

```
X_test = np.array(test_pizza_df[['diameter', 'n_topping']])
y_test = np.array(test_pizza_df['harga'])

print(f'X_test:\n{X_test}')
print(f'y_test: {y_test}')
print('\nBima Rizki')
```

```
X_test:
[[ 8  2]
 [ 9  0]
 [11  2]
 [16  2]
 [12  0]]
y_test: [11.   8.5 15.  18.  11. ]

Bima Rizki
```

2.3. Pengenalan *Multiple Linear Regression*

Multiple Linear Regression memungkinkan untuk menggunakan beberapa *explanatory variables* atau *feature*. Perbedaan antara *Simple Linear Regression* dengan *Multiple Linear Regression* terletak pada *feature* atau *explanatory variables* yang digunakan.

Formula *Multiple Linear Regression :*

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

```
[11]:
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print(f'r_squared: {r2_score(y_test, y_pred)}')
print('\nBima Rizki')
```

```
r_squared: 0.7701677731318468

Bima Rizki
```

2.4. Pengenalan *Polynomial Regression*

Polynomial Regression memodelkan hubungan antara *independent variable (feature) X* dan *dependent variable (target) y* sebagai derajat polynomial dalam *X*.

2.4.1. *Preprocessing Dataset*

```
[12]:
X_train = np.array(train_pizza_df['diameter']).reshape(-1, 1)
y_train = np.array(train_pizza_df['harga'])

print(f'X_train:\n{X_train}\n')
print(f'y_train: {y_train}')
print('\nBima Rizki')
```

```
X_train:
[[ 6]
 [ 8]
 [10]
 [14]
 [18]]

y_train: [ 7.   9.  13.  17.5 18. ]

Bima Rizki
```

2.5. Quadratic Polynomial Regression

[13]:

```
from sklearn.preprocessing import PolynomialFeatures

quadratic_feature = PolynomialFeatures(degree=2)
X_train_quadratic = quadratic_feature.fit_transform(X_train)

print(f'X_train_quadratic:\n{X_train_quadratic}\n')
```

```
X_train_quadratic:
[[  1.   6.  36.]
 [  1.   8.  64.]
 [  1.  10. 100.]
 [  1.  14. 196.]
 [  1.  18. 324.]]
```

[14]:

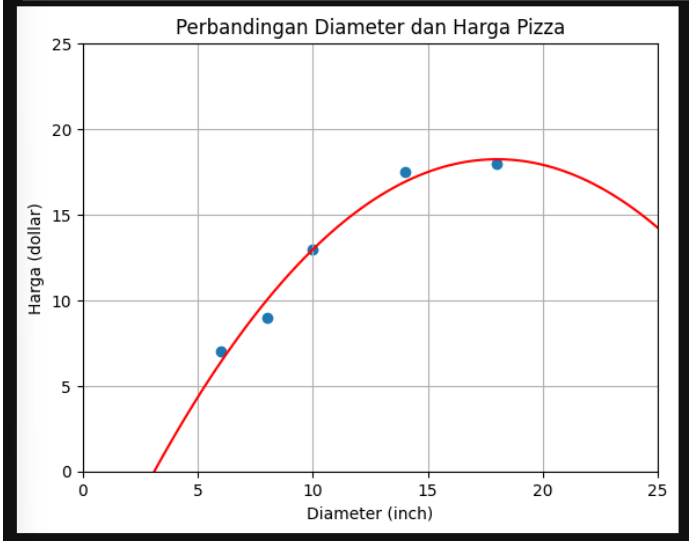
```
model = LinearRegression()
model.fit(X_train_quadratic, y_train)
```

[14]:

```
LinearRegression()
```

[17]:

```
import matplotlib.pyplot as plt

X_vis = np.linspace(0, 25, 100).reshape(-1, 1)
X_vis_quadratic = quadratic_feature.transform(X_vis)
y_vis_quadratic = model.predict(X_vis_quadratic)

plt.scatter(X_train, y_train)
plt.plot(X_vis, y_vis_quadratic, '-r')

plt.title('Perbandingan Diameter dan Harga Pizza')
plt.xlabel('Diameter (inch)')
plt.ylabel('Harga (dollar)')
plt.xlim(0, 25)
plt.ylim(0, 25)
plt.grid(True)
plt.show()
print('\nBima Rizki')
```

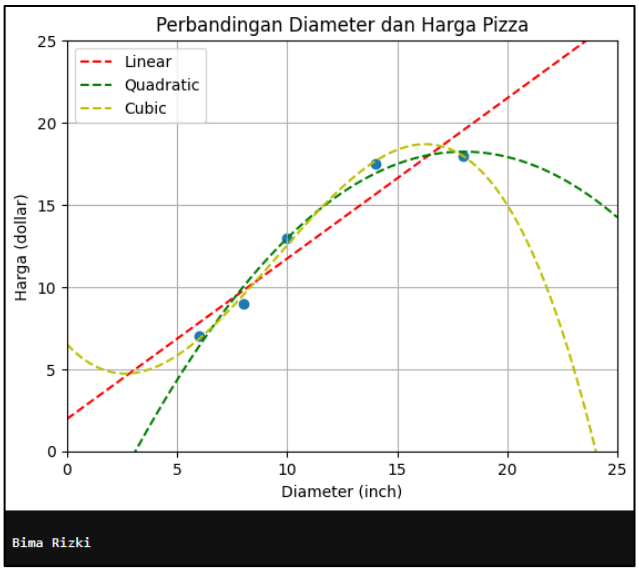



```
Bima Rizki
```

2.6. Linear Regression vs Quadratic Polynomial Regression vs Cubic Polynomial Regression

```
[20]: # Training Set
plt.scatter(X_train, y_train)

# Linear
model = LinearRegression()
model.fit(X_train, y_train)
X_vis = np.linspace(0, 25, 100).reshape(-1, 1)
y_vis = model.predict(X_vis)
plt.plot(X_vis, y_vis, '--r', label='Linear')

# Quadratic
quadratic_feature = PolynomialFeatures(degree=2)
X_train_quadratic = quadratic_feature.fit_transform(X_train)
model = LinearRegression()
model.fit(X_train_quadratic, y_train)
X_vis_quadratic = quadratic_feature.transform(X_vis)
y_vis = model.predict(X_vis_quadratic)
plt.plot(X_vis, y_vis, '--g', label='Quadratic')

# Cubic
cubic_feature = PolynomialFeatures(degree=3)
X_train_cubic = cubic_feature.fit_transform(X_train)
model = LinearRegression()
model.fit(X_train_cubic, y_train)
X_vis_cubic = cubic_feature.transform(X_vis)
y_vis = model.predict(X_vis_cubic)
plt.plot(X_vis, y_vis, '--y', label='Cubic')

plt.title('Perbandingan Diameter dan Harga Pizza')
plt.xlabel('Diameter (inch)')
plt.ylabel('Harga (dollar)')
plt.legend()
plt.xlim(0, 25)
plt.ylim(0, 25)
plt.grid(True)
plt.show()
print('\nBima Rizki')
```

Output dari kode di atas adalah sebagai berikut



3. *Logistic Regression* pada *Binary Classification Task*

3.1. Formula Dasar Pembentuk Logistic Regression | Fungsi Sigmoid

Formula *Simple Linear Regression*

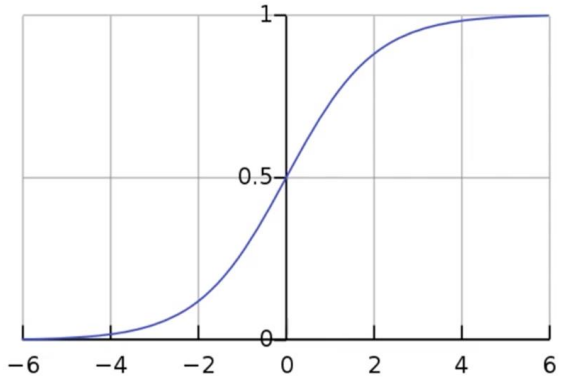
- $y = \alpha + \beta x$
- $g(x) = \alpha + \beta x$

Formula *Multiple Linear Regression*

- $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$
- $g(X) = \alpha + \beta X$

Formula *Logistic Regression*

- $g(X) = sigmoid(\alpha + \beta X)$
- $sigmoid(x) = \frac{1}{1+exp(-x)}$



3.2. Persiapan Dataset | SMS Spam Collection Dataset

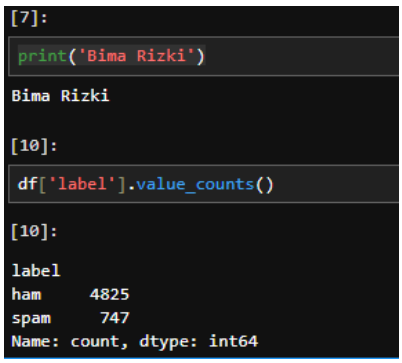
```
[6]:
import pandas as pd

df = pd.read_csv('./dataset/SMSSpamCollection',
                 sep='\t',
                 header=None,
                 names=['label', 'sms'])
df.head()
```

[6]:

	label	sms
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
[7]:
print('Bima Rizki')
```

```
Bima Rizki
```



```
[7]:
print('Bima Rizki')

Bima Rizki

[10]:
df['label'].value_counts()

[10]:

label
ham     4825
spam     747
Name: count, dtype: int64
```

3.3. Pembagian Training Dan Testing Set

```
[14]:
from sklearn.preprocessing import LabelBinarizer

X = df['sms'].values
y = df['label'].values

lb = LabelBinarizer()
y = lb.fit_transform(y).ravel()
lb.classes_

[14]:

array(['ham', 'spam'], dtype='<U4')
```

[18]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=0)
print(X_train, '\n')
print(y_train)
print('\nBima Rizki')
```

```
['Its going good...no problem..but still need little experience to understand american
customer voice...'
 'U have a secret admirer. REVEAL who thinks U R So special. Call 09065174042. To opt o
ut Reply REVEAL STOP. 1.50 per msg recd. Cust care 07821230901'
 'Ok...' ...
 "For ur chance to win a £250 cash every wk TXT: ACTION to 80608. T's&C's www.movietriv
ia.tv custcare 08712405022, 1x150p/wk"
 'R U &SAM P IN EACHOTHER. IF WE MEET WE CAN GO 2 MY HOUSE'
 'Mm feeling sleepy. today itself i shall get that dear']

[0 1 0 ... 1 0 0]

Bima Rizki
```

3.4. Feature Extraction Dengan TF-IDF

[28]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english')

X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

print(X_train_tfidf)
print('\nBima Rizki')
```

```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 32656 stored elements and shape (4179, 7287)>
  Coords        Values
  (0, 2997)     0.23173982975834367
  (0, 3007)     0.21421364306658514
  (0, 5123)     0.308974289326673
  (0, 4453)     0.2297719954323795
  (0, 3926)     0.3126721340000456
  (0, 2554)     0.3825278811525034
  (0, 6739)     0.3546359942830148
  (0, 900)      0.4114867709157148
  (0, 2006)     0.2898082580285881
  (0, 6903)     0.3591386422223876
  (1, 5642)     0.24344998442301355
  (1, 799)      0.25048918791028574
  (1, 5441)     0.5009783758205715
  (1, 6472)     0.24039776602646504
  (1, 6013)     0.20089911182610476
  (1, 216)      0.28902673040368515
  (1, 4677)     0.24039776602646504
  (1, 5394)     0.16464655071448758
  (1, 6131)     0.16142609035094446
  (1, 532)      0.20186022353306565
  (1, 4358)     0.17341410292348694
  (1, 5301)     0.2711077935907125
  (1, 2003)     0.2711077935907125
  (1, 1548)     0.18167737976542422
  (1, 36)       0.28902673040368515
  :     :
  (4176, 6782)  0.1407604617250961
```

```
  :     :
  (4176, 6792)  0.1407604617250961
  (4176, 6693)  0.16491299289150899
  (4176, 6684)  0.22114159453800114
  (4176, 7083)  0.19523751585154273
  (4176, 1569)  0.18895085073406012
  (4176, 7195)  0.17892283441772988
  (4176, 779)   0.2811068572055718
  (4176, 1612)  0.21138425595332702
  (4176, 365)   0.2388005587702937
  (4176, 7114)  0.4512018097459442
  (4176, 637)   0.29968668460649284
  (4176, 4350)  0.29968668460649284
  (4176, 2004)  0.25589560236817055
  (4176, 107)   0.29968668460649284
  (4176, 343)   0.2811068572055718
  (4177, 3319)  0.43046342221720785
  (4177, 4177)  0.3636187667918345
  (4177, 5565)  0.5506066649743346
  (4177, 2362)  0.6158854885899457
  (4178, 2068)  0.3055766821331892
  (4178, 2641)  0.3993042639531407
  (4178, 6555)  0.2897850627168302
  (4178, 5720)  0.3963527249882828
  (4178, 4279)  0.4530624713751054
  (4178, 5883)  0.548491137555895

Bima Rizki
```

3.5. Binary Classification Dengan Logistic Regression

```
[30]:
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train_tfidf, y_train)
y_pred = model.predict(X_test_tfidf)

for pred, sms in zip(y_pred[:5], X_test[:5]):
    print(f'PRED: {pred} - SMS: {sms}\n')

print('\nBima Rizki')
```

```
PRED: 0 - SMS: Storming msg: Wen u lift d phne, u say "HELLO" Do u knw wt is d real mea
ning of HELLO?? . . . It's d name of a girl..! . . . Yes.. And u knw who is dat girl??
"Margaret Hello" She is d girlfrnd f Grahmbell who invnted telphone... . . . . Moral:On
e can 4get d name of a person, bt not his girlfrnd... G o o d n i g h t . . .@

PRED: 0 - SMS: <Forwarded from 448712404000>Please CALL 08712404000 immediately as ther
e is an urgent message waiting for you.

PRED: 0 - SMS: And also I've sorta blown him off a couple times recently so id rather n
ot text him out of the blue looking for weed

PRED: 0 - SMS: Sir Goodmorning, Once free call me.

PRED: 0 - SMS: All will come alive.better correct any good looking figure there itsel
f..


Bima Rizki
```

3.6. Evaluation Metrics Pada Binary Classification Task

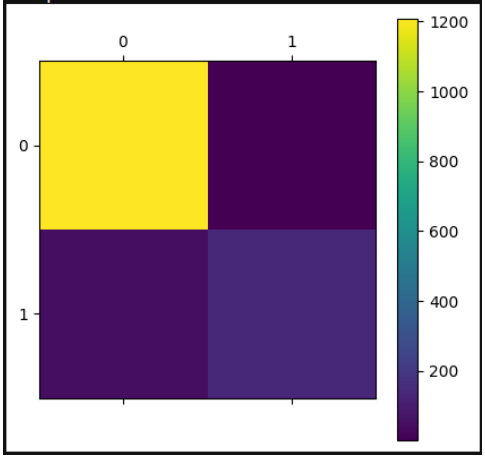
Terminologi atau istilah dasar:

- *True Positive (TP)*
- *True Negative (TN)*
- *False Positive (FP)*
- *False Negative(FN)*

3.6.1. Pengenalan Confusion Matrix

```
[31]:
from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_test, y_pred)
matrix

[31]:

array([[1207,    1],
       [  47,  138]])

[32]:
tn, fp, fn, tp = matrix.ravel()

print(f'TN: {tn}')
print(f'FP: {fp}')
print(f'FN: {fn}')
print(f'TP: {tp}')
print('\nBima Rizki')

TN: 1207
FP: 1
FN: 47
TP: 138

Bima Rizki
```

```
Bima Rizki

[33]:
import matplotlib.pyplot as plt

plt.matshow(matrix)
plt.colorbar()

[33]:

<matplotlib.colorbar.Colorbar at 0x1e530ccd250>
```



3.6.2. Pengenalan Accuracy Score

Accuracy mengukur porsi dari hasil prediksi yang tepat.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{correct}{total}$$

```
[34]:
from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_pred)

[34]:
0.9655419956927495
```

Nilai pada gambar diatas termasuk akurasi tingkat tinggi, karena jika diubah menjadi persentasi menjadi 96,48 %.

3.6.3. Pengenalan Precision Dan Recall

Selain menggunakan *accuracy*, performa dari suatu *classifier* umumnya juga diukur berdasarkan nilai *Precission* dan *Recall*.

Precission and Positive Predictive Value (PPV)

$$Precission = \frac{TP}{TP+FP}$$

```
[35]:
from sklearn.metrics import precision_score

precision_score(y_test, y_pred)

[35]:
np.float64(0.9928057553956835)
```

Recall or True Positive Rate (TPR) or Sensitivity

$$Recall = \frac{TP}{TP+FN}$$

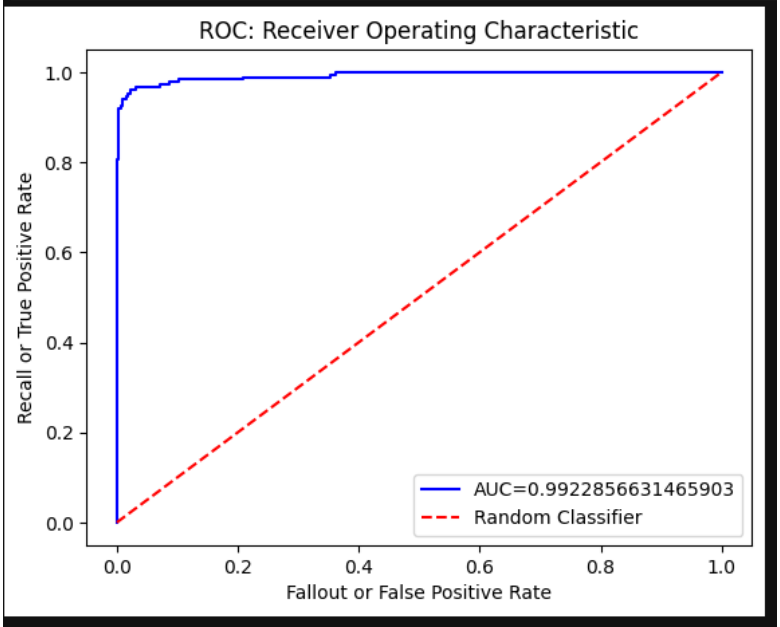
```
[36]:
from sklearn.metrics import recall_score

recall_score(y_test, y_pred)

[36]:
np.float64(0.745945945945946)
```

3.6.4. Pengenalan F1 Score | F1 Measure

```
[37]:
from sklearn.metrics import f1_score

f1_score(y_test, y_pred)

[37]:
np.float64(0.8518518518518519)
```

3.6.5. Pengenalan ROC | Receiver Operating Characteristic

```
[39]:
from sklearn.metrics import roc_curve, auc

prob_estimates = model.predict_proba(X_test_tfidf)

fpr, tpr, threshhold = roc_curve(y_test, prob_estimates[:, 1])
nilai_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, 'b', label=f'AUC={nilai_auc}')
plt.plot([0,1], [0,1], 'r--', label='Random Classifier')

plt.title('ROC: Receiver Operating Characteristic')
plt.xlabel('Fallout or False Positive Rate')
plt.ylabel('Recall or True Positive Rate')
plt.legend()
plt.show()
print('\nBima Rizki')
```




```
Bima Rizki
```