Nama : Suci Maria

Kelas : TIF-A1

Npm : 41155050210005

TUGAS PERTEMUAN 2

1. **Lakukan praktek dari https://youtu.be/lcjq7-2zMSA?si=f4jWJR6lY8y0BZKl dan buat screen shot hasil run dengan nama anda pada hasil run tersebut. Praktek tersebut yaitu:**

1.1. Sample dataset

```
[1]: 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
```

[1]:

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

```
[2]: print("\nNama: Suci Maria")
```

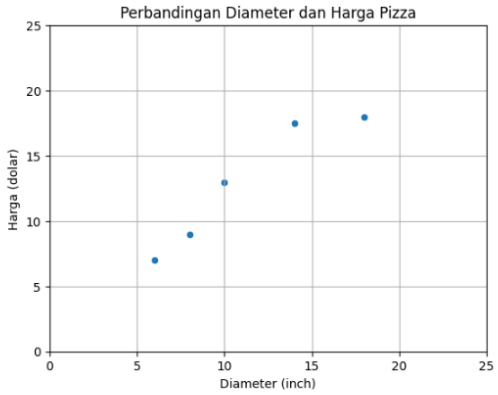
```
Nama: Suci Maria
```

1.2. Visualisasi dataset

```
[4]: 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 (dolar)')
plt.xlim(0, 25)
plt.ylim(0, 25)
plt.grid(True)
plt.show()
```




```
[5]: print("\nNama: Suci Maria")
```

```
Nama: Suci Maria
```

1.3. Transformasi dataset

```
[7]: import numpy as np

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

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

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

```
[8]: print("\nNama: Suci Maria")
```

```
Nama: Suci Maria
```

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

```
[9]: (5, 1)
```

```
[10]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

```
[11]: X
```

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

```
[12]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

1.4. Training Simple Linear Regression Model

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

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

```
[13]: LinearRegression
      LinearRegression()
```

```
[14]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

1.5. Visualisasi Simple Linear Regression Model | Penjelasan persamaan garis linear

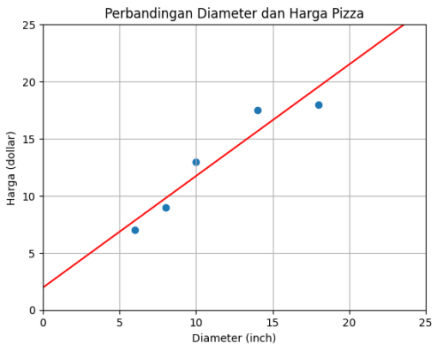
```
[15]: X_vis = np.array([0, 25]).reshape(-1, 1)
      y_vis = model.predict(X_vis)
```

```
[16]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

```
[17]: 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()
```



Formula Linear Regression: $y = \alpha + \beta x$

- y : response variable
- x : explanatory variable
- α : interecpt
- β : slope

```
[18]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

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

      intercept: 1.965517241379315
      slope: [0.9762931]

[20]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

1.6. Kalkulasi nilai slope

- **Mencari Nilai slope**

  Nilai slope pada Linear Regression bisa diperoleh dengan memanfaatkan formula berikut:

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

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

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

      X flatten: [ 6  8 10 14 18]

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

[23]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

- **Variance**

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

      print(f'variance: {variance_x}')

      variance: 23.2

[25]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

- **Covariance**

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

[26]: array([[23.2 , 22.65],
             [22.65, 24.3 ]])

[27]: print("Nama: Suci Maria")

      Nama: Suci Maria

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

      print(f'covariance: {covariance_xy}')

      covariance: 22.650000000000002

[29]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

- **Slope**

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

      print(f'slope: {slope}')

      slope: 0.976293103448276

[31]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

1.7. Kalkukasi nilai intercept

- Nilai intercept pada Linear Regression bisa diperoleh dengan meamnfaatkan formula berikut:

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

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

      print(f'intercept: {intercept}')

      intercept: 1.9655172413793096

[33]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

1.8. Prediksi harga pizza dengan Simple Linear Regression Model

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

[34]: array([[12],
             [20],
             [23]])

[35]: print("Nama: Suci Maria")

      Nama: Suci Maria

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

[37]: array([13.68103448, 21.49137931, 24.42025862])

[38]: print("Nama: Suci Maria")

      Nama: Suci Maria

[39]: for dmtr, hrg in zip(diameter_pizza, prediksi_harga):
          print(f'Diameter: {dmtr} prediksi harga: {hrg}')

      Diameter: [12] prediksi harga: 13.681034482758621
      Diameter: [20] prediksi harga: 21.491379310344826
      Diameter: [23] prediksi harga: 24.42025862068965

[40]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

1.9. Evaluasi model dengan Coefficient of Determination | R Squared

➢ Training & Testing Dataset

```
[44]: 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])

[45]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

➢ Trining Simple Linear Regression Model

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

[46]: LinearRegression()

[47]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

➢ Evaluasi Linear Regression Model dengan Cofficient of Determination atau R-squared(R^2)

```
[49]: 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}')

      R-squared: 0.662005292942553

[50]: print("Nama: Suci Maria")

      Nama: Suci Maria
```

1.10. Kalkulasi nilai R Squared | Coefficient of Determination

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

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

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

SS_{res}

```
[53]: 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}')
```

```
ss_res: 19.1980993608799
```

```
[54]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

SS_{tot}

```
[55]: 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}')
```

```
ss_tot: 56.8
```

```
[56]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

R^2

```
[57]: r_squared = 1 - (ss_res / ss_tot)

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

```
R-squared: 0.6620052929422553
```

```
[58]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

2. Lakukan praktek dari https://youtu.be/nWJUJenAyB8?si=BQDzWwrMnr8jtzpV dan buat screen shot hasil run dengan nama anda pada hasil run tersebut. Praktek tersebut yaitu:

2.1. Persiapan sample dataset

- **Training Dataset**

```
[1]: import pandas as pd

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

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

[1]:

	diameter	n_toping	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]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

- **Testing Dataset**

```
[3]: import pandas as pd

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

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

[3]:

	diameter	n_toping	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("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

2.2. Preprocessing dataset

- **Training**

```
[5]: import numpy as np

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

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

```
X_train:
[[ 6  2]
 [ 8  1]
 [10  0]
 [14  2]
 [18  0]]

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

```
[6]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

- **Testing**

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

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

```
X_test:
[[ 8  2]
 [ 9  0]
 [11  2]
 [16  2]
 [12  0]]

y_test:[11.   8.5 15.  18.  11. ]
```

```
[8]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

2.3. Pengenalan Multiple Linear Regression | Apa itu Multiple Linear Regression?

Multiple Linaear Regression merupakan generalisasi dari simple Linear Regression yang memungkinkan untuk menggunakan beberapa explanatory variables.

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

```
[9]: 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)}')
```

```
r_squared: 0.7701677731318468
```

```
[10]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

2.4. Pengenalan Polynomial Regression | Apa itu Polynomial Regression?

Polynomial Regression memodelkan hubungan antara independent variable x dan dependent variable y sebagai derajat polynomial dalam x.

- **Preprocessing Dataset**

```
[11]: 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}')
```

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

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

```
[12]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

2.5. Quadratic Polynomial Regression

$$y = \alpha + \beta_1 x + \beta_2 x^2$$

➢ Polynomial Features

```
[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]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

➢ Training Model

```
[15]: model = LinearRegression()
model.fit(X_train_quadratic, y_train)
```

```
[15]: LinearRegression
LinearRegression()
```

```
[16]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

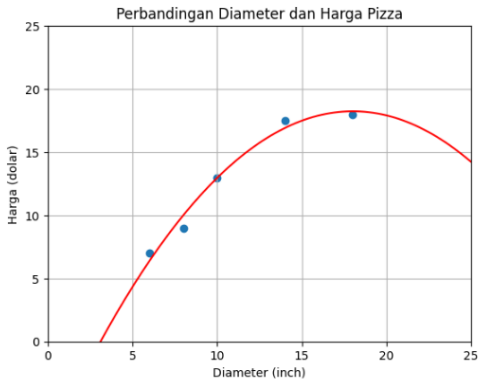
➢ Visualisasi Model

```
[22]: 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 (dolar)')
plt.xlim(0, 25)
plt.ylim(0, 25)
plt.grid(True)
plt.show()
```




```
[23]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

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

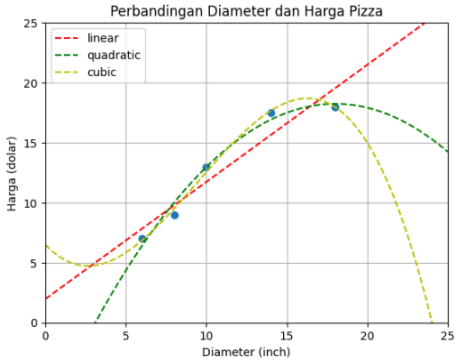
```
[24]: # Training Set
plt.scatter(X_train, y_train)

# 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 = 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 (dolar)')
plt.legend()
plt.xlim(0, 25)
plt.ylim(0, 25)
plt.grid(True)
plt.show()
```




```
[25]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

3. Lakukan praktek dari https://youtu.be/oe7DW4rSH1o?si=H-PZJ9rs9-Kab-Ln dan buat screen shot hasil run dengan nama anda pada hasil run tersebut. Praktek tersebut yaitu:

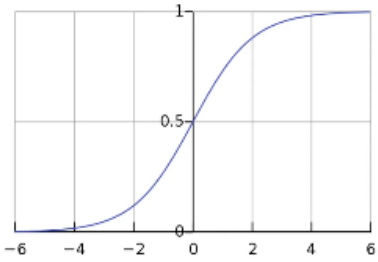
3.1. Formula dasar pembentuk Logistic Regression | Fungsi Sigmoid

Formula dasar

- **Simple Linear Regression**
  - $y = \alpha + \beta x$
  - $g(x) = \alpha + \beta x$
- **Multiple Linear Regression**
  - $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$
  - $g(X) = \alpha + \beta X$

➢ **Logistic Regression**

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



3.2. Persiapan dataset | SMS Spam Collection Dataset

```
[1]: import pandas as pd

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

     df.head()
```

[1]:

	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...

```
[2]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

```
[4]: print("Nama: Suci Maria")
```

```
[3]: df
```

```
[3]: df.info()
```

3.3. Pembagian training dan testing set

```
[6]: print("Nama: Suci Maria")
```

```
[2]: X = df.sms.values
     y = df.label_enc.values
```

```
[7]: 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)
```

```
['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 out 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.movietrivia.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]
```

```
[8]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

3.4. Feature extraction dengan TF-IDF

```
[8]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

```
[10]: 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)
```

```
<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)     0.243449984423013355
  (1, 799)      0.25048918791028574
  (1, 5441)     0.50097837582057l5
  (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, 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
```

3.5. Binary Classification dengan Logistic Regression

```
[12]: 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')
```

```
PRED: 0 - SMS: Storming msg: Wen u lift d phne, u say "HELLO" Do u knw wt is d real meaning of HELLO?? . . . It's d name of a gir
l..! . . . 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 there 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 not 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 itself..
```

```
[13]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

3.6. Evaluation Metrics pada Binary Classification Task

- **Terdiri dari beberapa matrix diantaranya :**
  - Confusion Matrix
  - Accuracy
  - Precision dan Recall
  - F1 Score

- ROC

➢ **Terminologi Dasar**

- **True Positive (TP)**

  Sesuatu yang bernilai Positif telah dengan tepat diprediksi sebagai Positif oleh model, contoh : model sudah dengan tepat memprediksi data spam sebagai spam dan data ham sebagai ham.

- **True Negative (TN)**

  Sesuatu yang bernilai Negatif telah dengan tepat diprediksi sebagai Negatif oleh model, contoh : model sudah dengan tepat memprediksi data ham sebagai bukan spam dan data spam sebagai bukan ham.

- **False Positive (FP)**

  Sesuatu yang bernilai Negatif telah keliru di prediksi sebagai Positif oleh model, contoh : model sudah dengan keliru memprediksi data ham sebagai spam dan data spam sebagai ham.

- **False Negative (FN)**

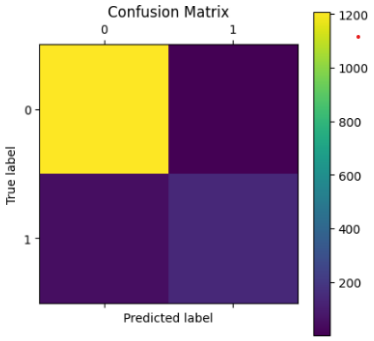
  Sesuatu yang bernilai Positif telah keliru di prediksi sebagai Negatif oleh model, contoh : model sudah dengan keliru memprediksi data spam sebagai bukan spam dan data ham sebagai bukan ham.

3.7. Pengenalan Confusion Matrix

```
[19]: import matplotlib.pyplot as plt

      plt.matshow(matrix)
      plt.colorbar()

      plt.title('Confusion Matrix')
      plt.ylabel('True label')
      plt.xlabel('Predicted label')
      plt.show()
```




```
[20]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

3.8. Pengenalan Accuracy Score

Accuracy mengukur porsi dari hasil prediksi yang tepat.

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

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

      accuracy_score(y_test, y_pred)
```

```
[21]: 0.9655419956927495
```

```
[22]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

3.9. Pengenalan Precision dan Recall

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

- ***Precission or Positive Predictive Value (PPV)***

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

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

      precision_score(y_test, y_pred)
```

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

```
[24]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

- ***Recall or True Positive Rate (TPR) or Sensitivity***

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

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

      recall_score(y_test, y_pred)
```

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

```
[26]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

3.10. Pengenalan F1 Score | F1 Measure

F1-score atau F1-measure adalah harmonic mean dari precission dan recall

$$F1\ score = \frac{precission \times recall}{precission + recall}$$

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

      f1_score(y_test, y_pred)
```

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

```
[28]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```

3.11. Pengenalan ROC | Receiver Operating Characteristic

ROC menawarkan visualisasi terhadap performa dari classifier dengan membandingkan nilai Recall(TPR) dan nilai Fallout(FPR)

$$fallout = \frac{FP}{TN+FP}$$

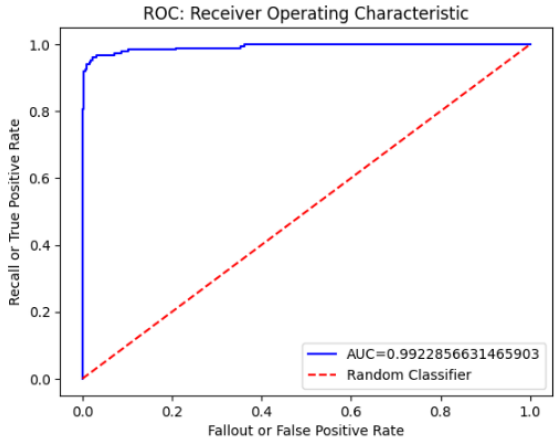
```
[29]: from sklearn.metrics import roc_curve, auc

      prob_estimates = model.predict_proba(X_test_tfidf)

      fpr, tpr, threshold = 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()
```




```
[30]: print("Nama: Suci Maria")
```

```
Nama: Suci Maria
```