Tugas Pertemuan 2

Nama : Rizky muhamad wicaksana

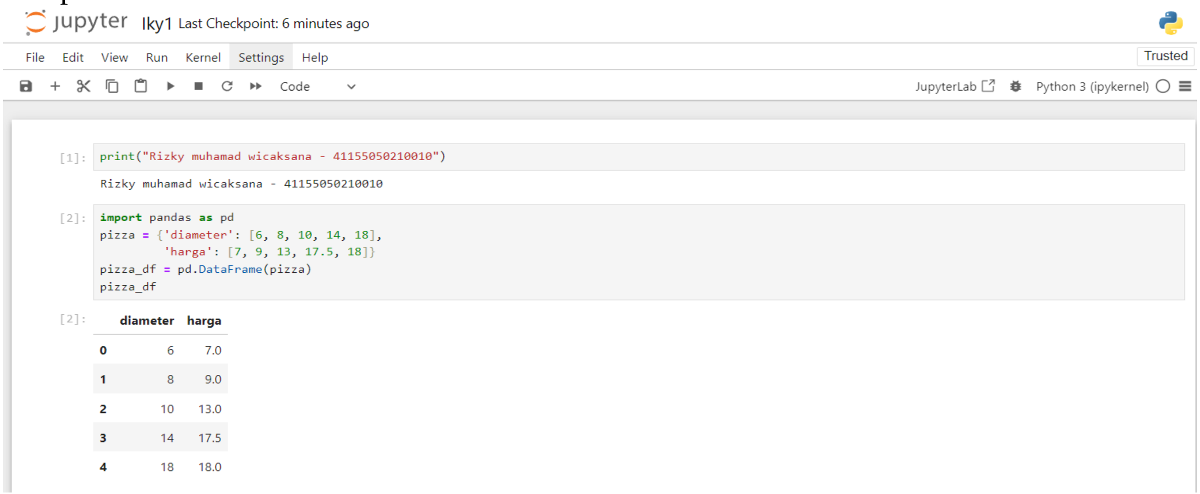
NPM : 41155050210010

Informatika A1

Machine Learning

1.0. Simple Linear Regression

Sample Dataset

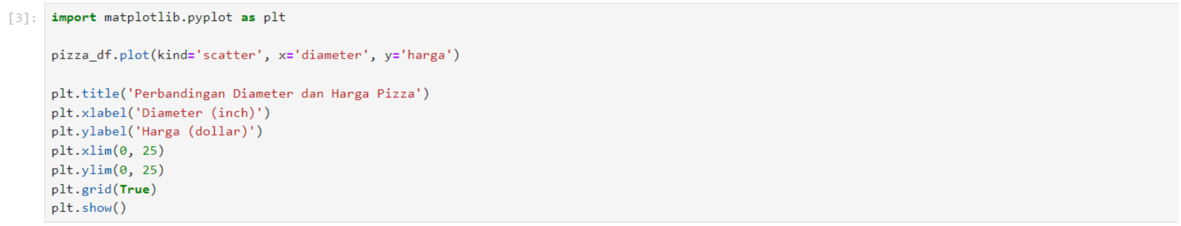


```
[1]: print("Rizky muhamad wicaksana - 41155050210010")

     Rizky muhamad wicaksana - 41155050210010

[2]: 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
```

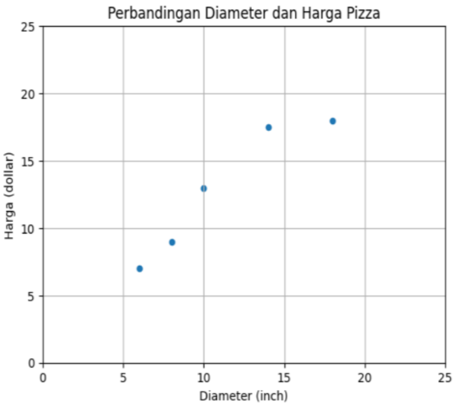
[2]:

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

Visualisasi Dataset



```
[3]: 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()
```



Penyesuaian Data

[4]:

```
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. ]
```

[5]:

```
x = x.reshape(-1, 1)
x.shape
```

[5]:

```
(5, 1)
```

[6]:

```
x
```

[6]:

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

Training Simple Linear Regression Model

[7]:

```
from sklearn.linear_model import LinearRegression

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

[7]:

```
▾  LinearRegression
LinearRegression()
```

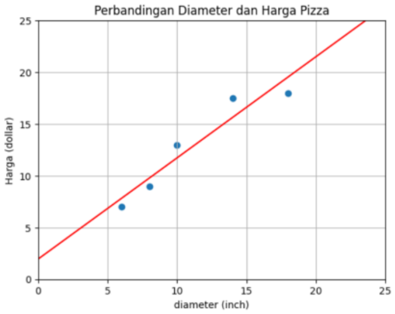
Visualisasi Simple Linear Regression Model

[8]:

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

[9]:

```
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()
```



[10]:

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

```
intercept: 1.965517241379315
slope: [0.9762931]
```

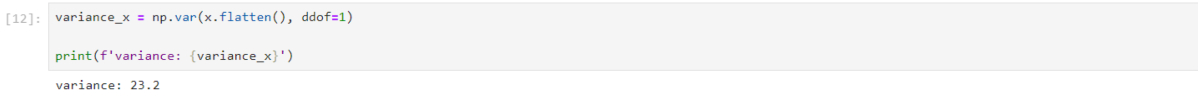
Mencari Nilai Slope

[11]:

```
print(f'x:\n{x}\n')
print(f'x flatten: {x.flatten()}\n')
print(f'y: {y}')
```

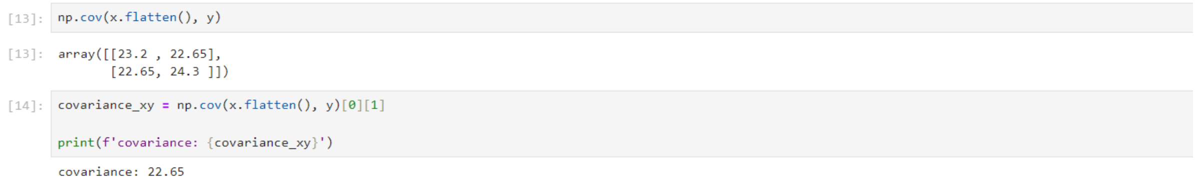
```
x:
[[ 6]
 [ 8]
 [10]
 [14]
 [18]]

x flatten: [ 6  8 10 14 18]

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

Variance



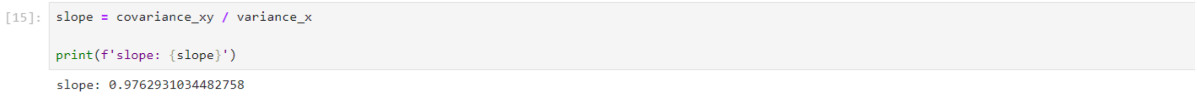
```
[12]: variance_x = np.var(x.flatten(), ddof=1)

      print(f'variance: {variance_x}')

      variance: 23.2
```

Covariance



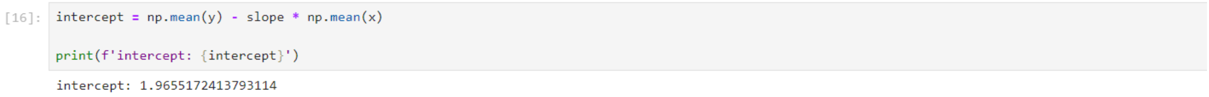
```
[13]: np.cov(x.flatten(), y)

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

[14]: covariance_xy = np.cov(x.flatten(), y)[0][1]

      print(f'covariance: {covariance_xy}')

      covariance: 22.65
```

Slope



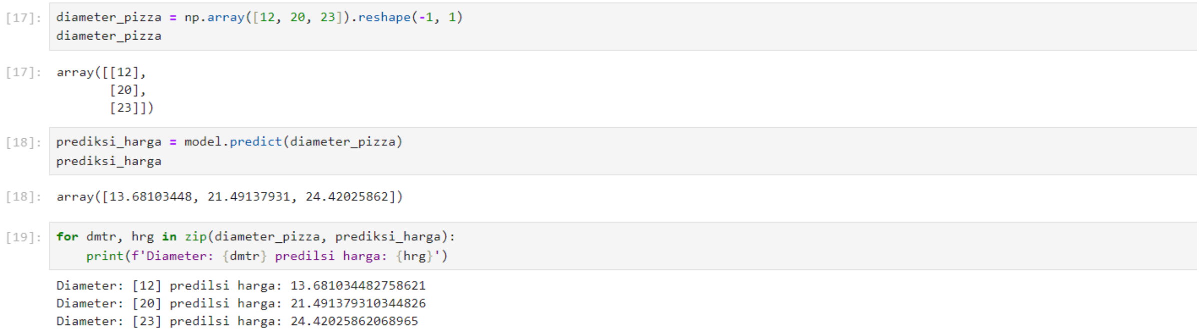
```
[15]: slope = covariance_xy / variance_x

      print(f'slope: {slope}')

      slope: 0.9762931034482758
```

Mencari Nilai Intercept



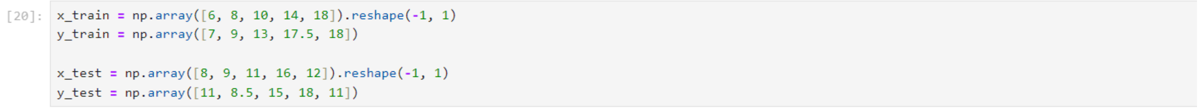
```
[16]: intercept = np.mean(y) - slope * np.mean(x)

      print(f'intercept: {intercept}')

      intercept: 1.9655172413793114
```

Prediksi Harga Pizza



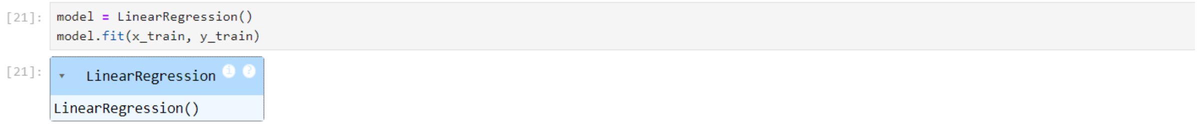
```
[17]: diameter_pizza = np.array([12, 20, 23]).reshape(-1, 1)
      diameter_pizza

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

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

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

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

      Diameter: [12] predilsi harga: 13.681034482758621
      Diameter: [20] predilsi harga: 21.491379310344826
      Diameter: [23] predilsi harga: 24.42025862068965
```

Training & testing data



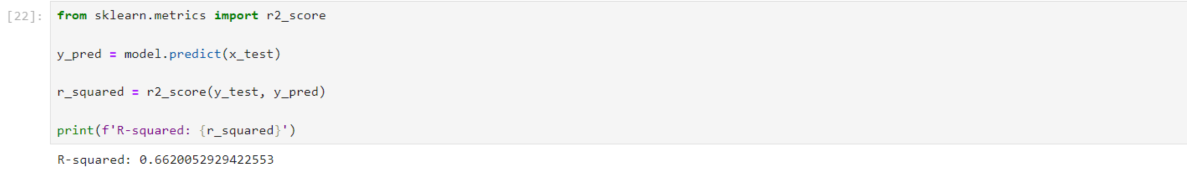
```
[20]: x_train = np.array([6, 8, 10, 14, 18]).reshape(-1, 1)
      y_train = np.array([7, 9, 13, 17.5, 18])

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

Training simple Linear Regression Model



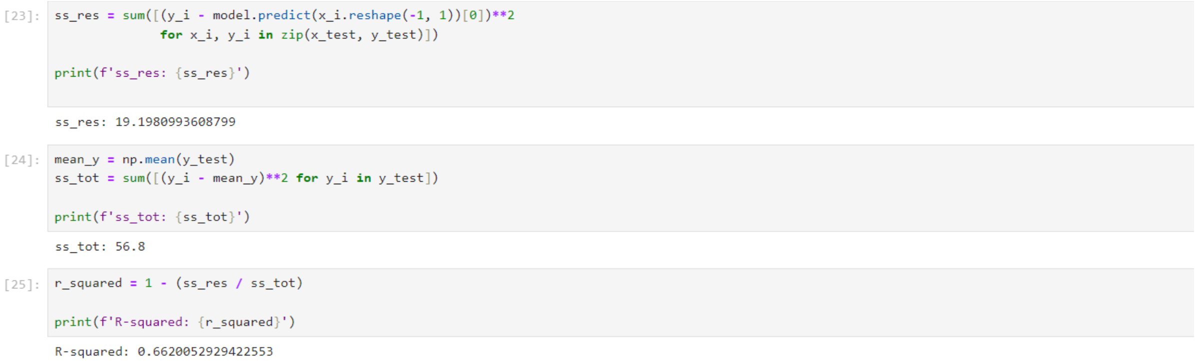
```
[21]: model = LinearRegression()
      model.fit(x_train, y_train)

[21]: LinearRegression
      LinearRegression()
```

Evaluasi Linear Regression Model dengan Coefficient of Determination R-squared



```
[22]: 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: 0.6620052929422553
```

Kalkulasi nilai R Squared | Coefficient of Determination



[23]:

```
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
```

[24]:

```
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}')
```

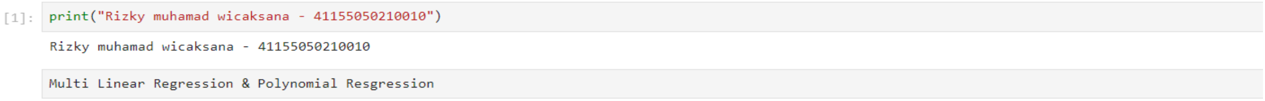
```
ss_tot: 56.8
```

[25]:

```
r_squared = 1 - (ss_res / ss_tot)

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

```
R-squared: 0.6620052929422553
```

2.0. Multi Linear Regression & Polynomial Resgression



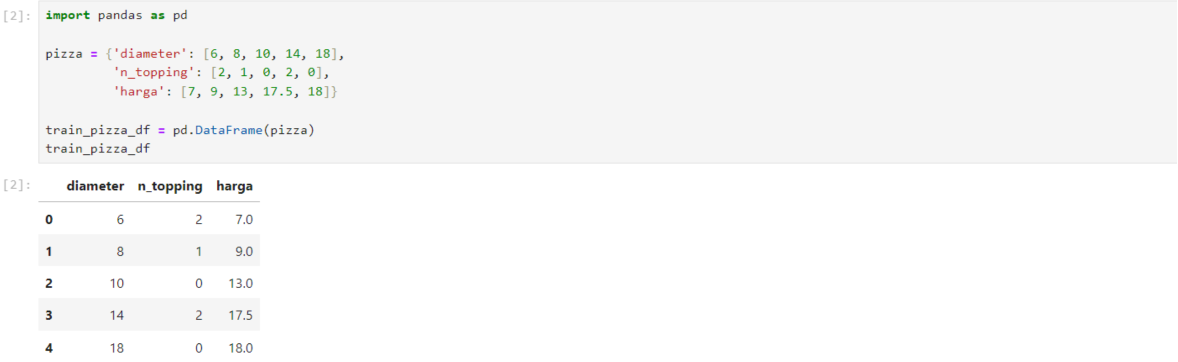
[1]:

```
print("Rizky muhamad wicaksana - 41155050210010")
```

```
Rizky muhamad wicaksana - 41155050210010
```

Multi Linear Regression & Polynomial Resgression

Persiapan sample dataset
Training dataset



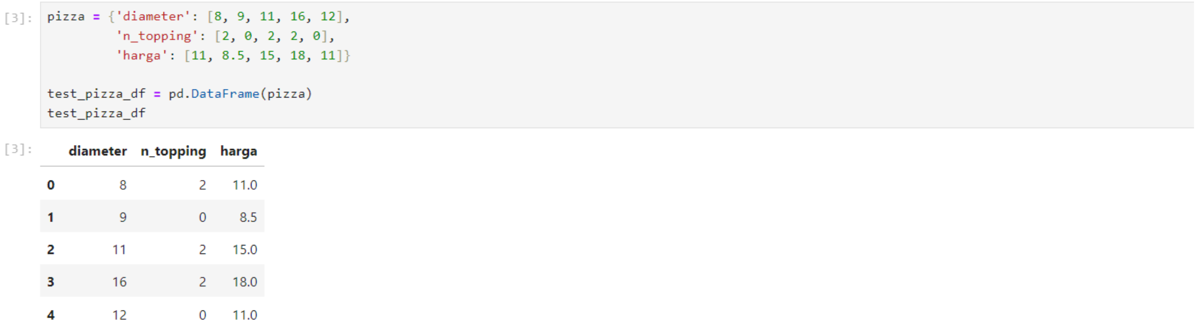
[2]:

```
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
```

[2]:

	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

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

Preprocessing Dataset

[4]:

```
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}\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. ]
```

[5]:

```
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}\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. ]
```

Multi Linear Regression

[6]:

```
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
```

Polynomial Regression

Preprocessing Dataset

[7]:

```
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. ]
```

Polynomial Regression: Quadratic

Polynomial Features

[8]:

```
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.]]
```

Training Model

[9]:

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

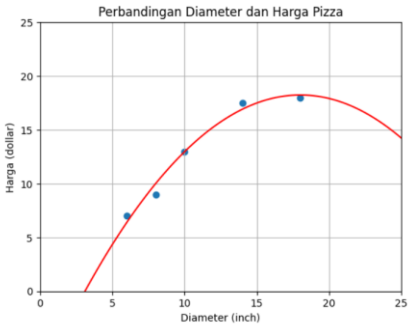
[9]:

```
▾ LinearRegression
LinearRegression()
```

Visualisasi Model

[11]:

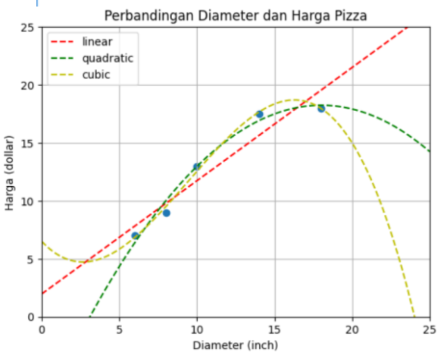
```
import matplotlib.pyplot as plt

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()
```



Polynomial Regression: Quadratic vs Cubic

[13]:

```
# 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()
```



3.0. Logistic Regression pada Binary Classification Task

Dataset: SMS Spam Collection Data set

[1]:

```
print("Rizky muhamad wicaksana - 41155050210010")
```

```
Rizky muhamad wicaksana - 41155050210010
```

```
Logistic Regression pada Binary Classification Task
```

[10]:

```
import pandas as pd

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

[10]:

	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...

[11]:

```
df['label'].value_counts()
```

[11]:

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

Training & Testing Dataset

[12]:

```
from sklearn.preprocessing import LabelBinarizer
x = df['sms'].values
y = df['label'].values

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

[12]:

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

[13]:

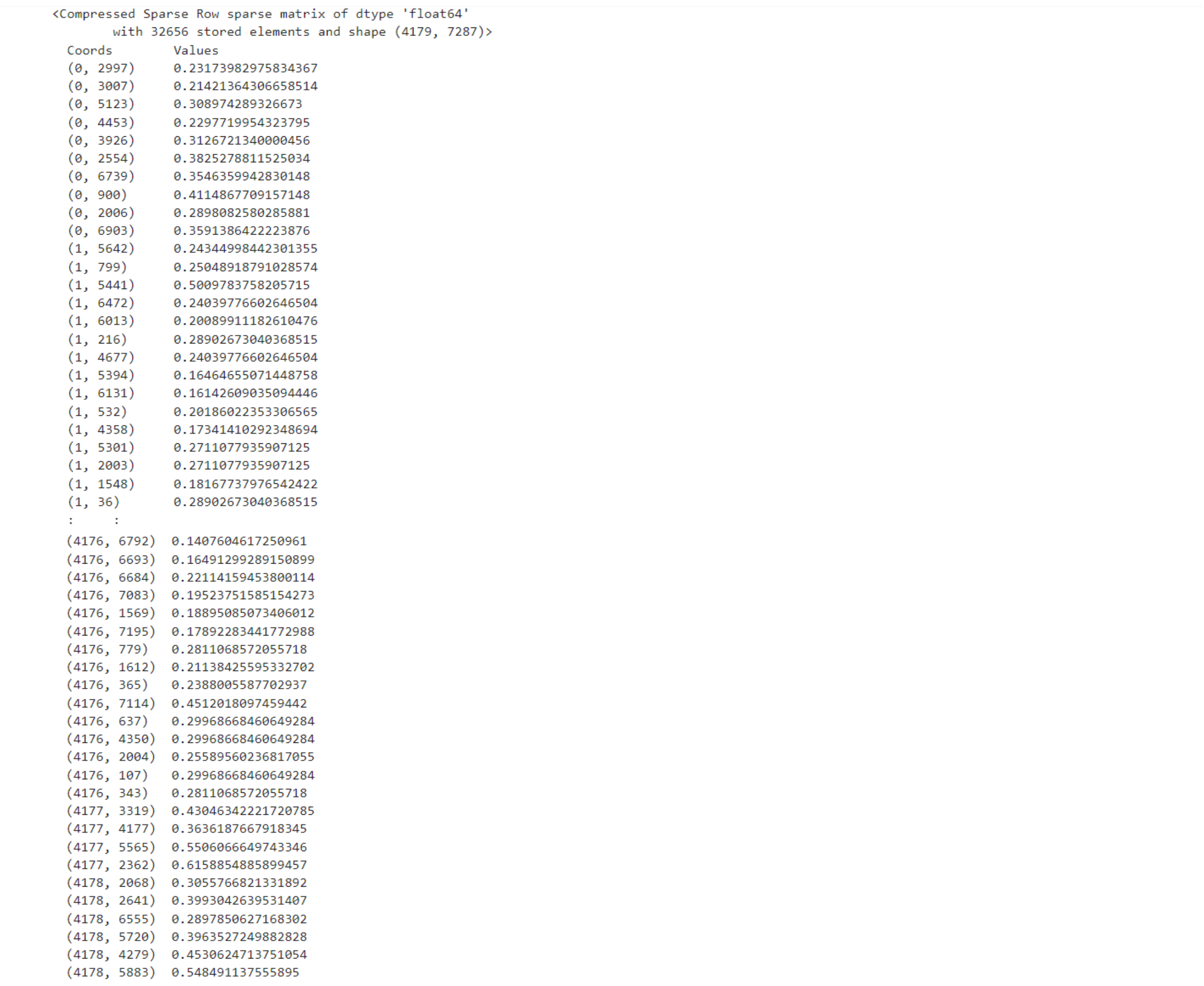
```
from sklearn.model_selection import train_test_split

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]
```

Feature Extraction dengan TF-IDF

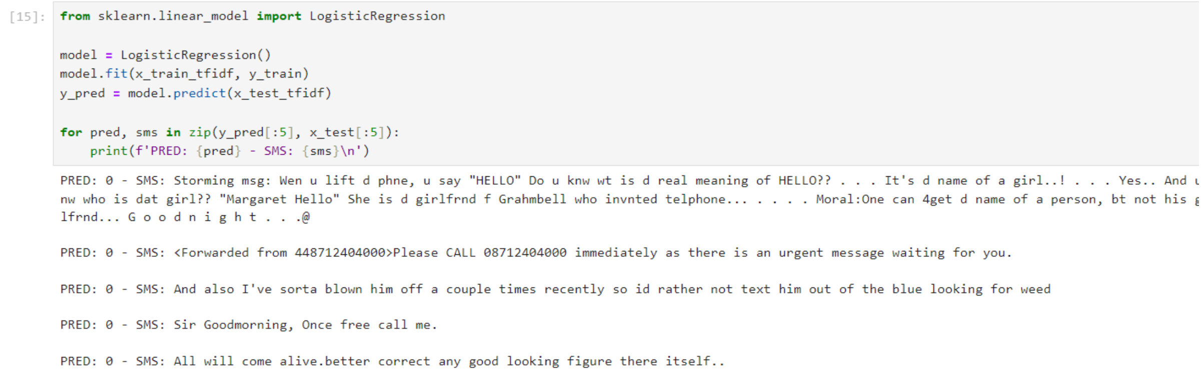
[14]:

```
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)	0.24344998442301355
  (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, 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
```

Binary Classification dengan Logistic Regression



```
[15]: 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 girl..! . . . Yes.. And u
nw who is dat girl?? "Margaret Hello" She is d girlfrnd f Grahmbell who invnted telphone... . . . . Moral:One can 4get d name of a person, bt not his g
lfrnd... 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..
```

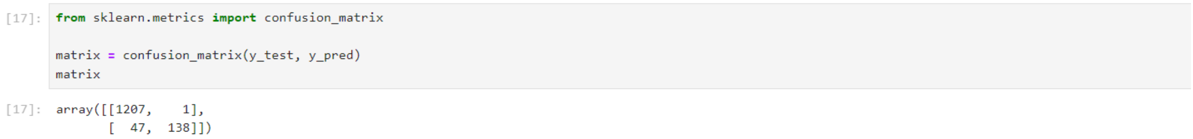
Evaluation Metrics pada Binary Classification

- Confusion Matrix
- Accuracy
- Precission & Recall
- F1Score
- ROC

Terminologi Dasar

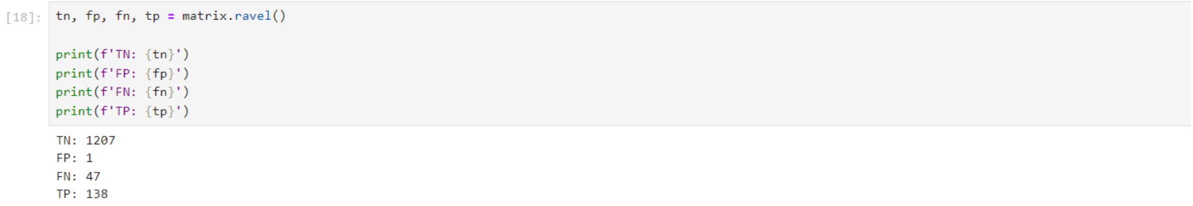
- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

Confusion Matrix



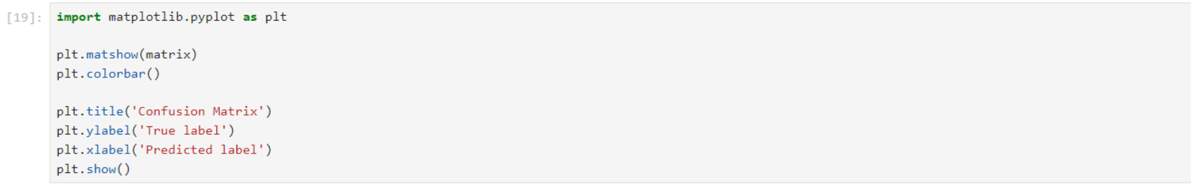
```
[17]: from sklearn.metrics import confusion_matrix

      matrix = confusion_matrix(y_test, y_pred)
      matrix
```

```
[17]: array([[1207,    1],
             [  47,  138]])
```

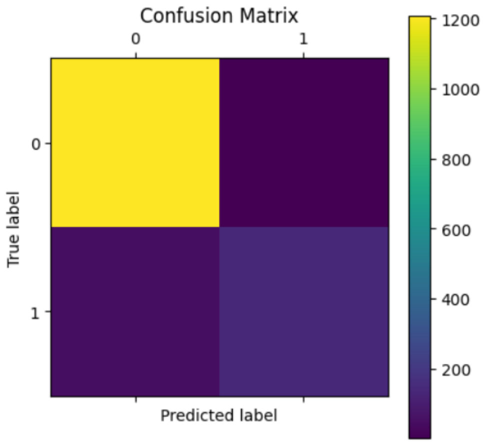


```
[18]: tn, fp, fn, tp = matrix.ravel()

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

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



```
[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()
```



Accuracy

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

      accuracy_score(y_test, y_pred)
```

```
[20]: 0.9655419956927495
```

Precission & Recall

Precission or Positive Predictive Value (PPV)

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

      precision_score(y_test, y_pred)
```

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

Recall or True Positive Rate (TPR) or Sensitivity

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

      recall_score(y_test, y_pred)
```

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

F1 score

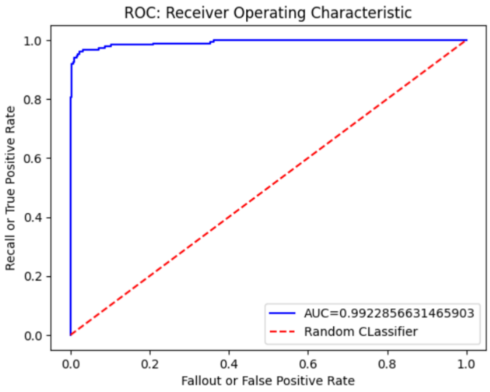
Harmonic mean dari precission dan recall

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

      f1_score(y_test, y_pred)
```

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

ROC: Receiver Operating Characteristic

```
[25]: 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()
```


ROC: Receiver Operating Characteristic
Recall or True Positive Rate
Fallout or False Positive Rate
AUC=0.9922856631465903
Random CLassifier