# MACHINE LEARNING

Disusun Oleh:

Nama : Ilpan

NPM : 41155050210046

Kelas : A – 2

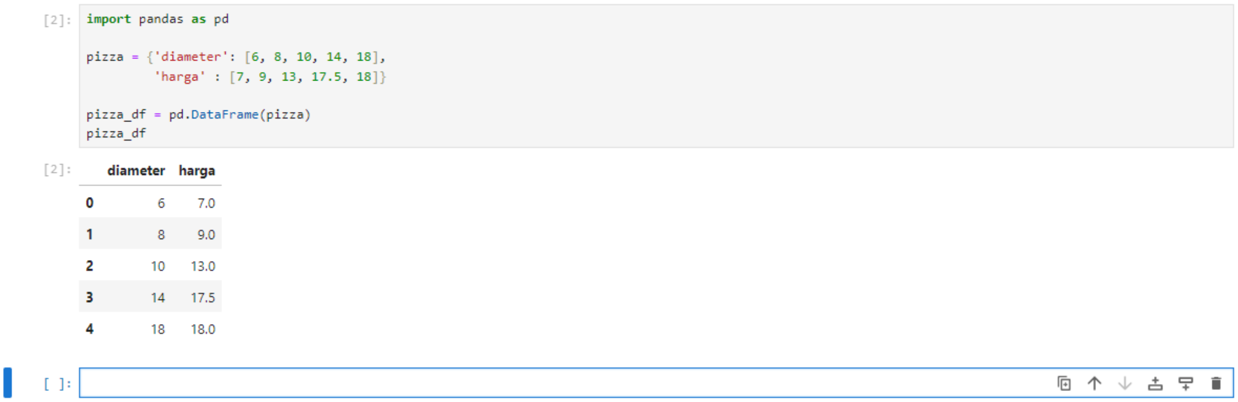
**TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS LANGLANGBUANA**

**2024**

## Sample dataset

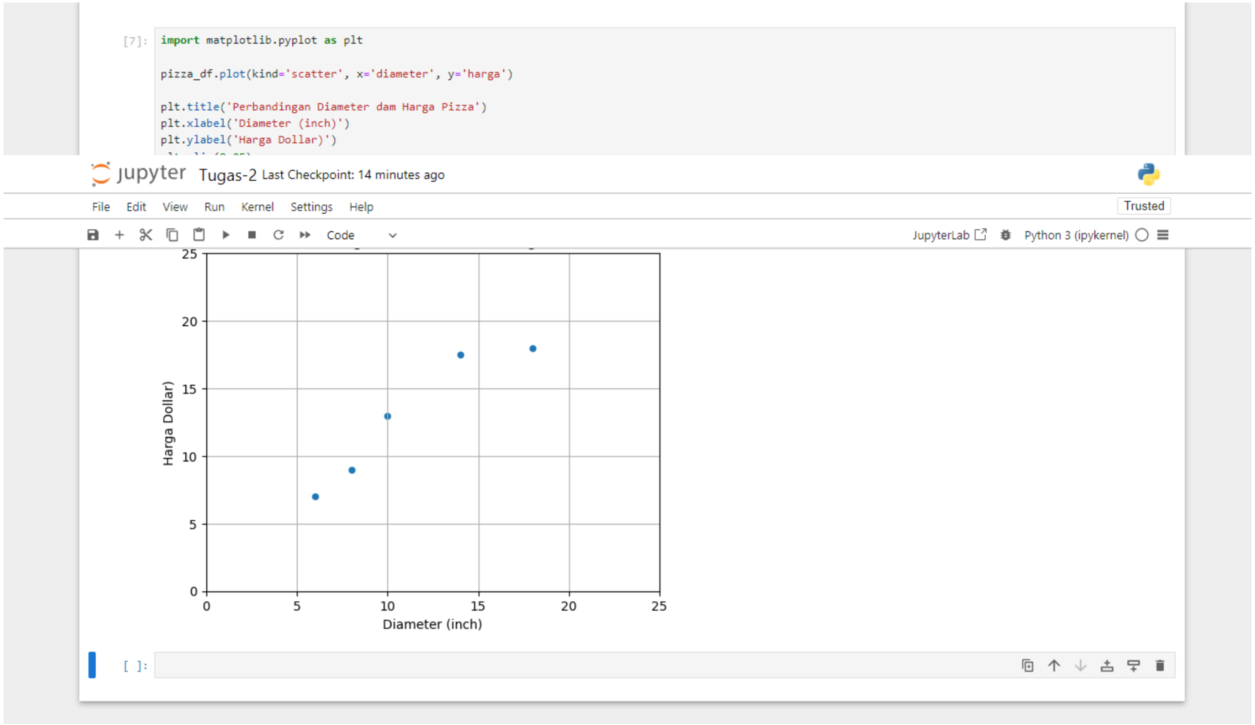


```
[2]: import pandas as pd

     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



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

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

     plt.title('Perbandingan Diameter dam Harga Pizza')
     plt.xlabel('Diameter (inch)')
     plt.ylabel('Harga Dollar)')
```

## Transformasi dataset

```
[9]: 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. ]
```

```
[12]: x = x.reshape(-1, 1)
      x.shape

[12]: (5, 1)
```

```
[13]: x

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

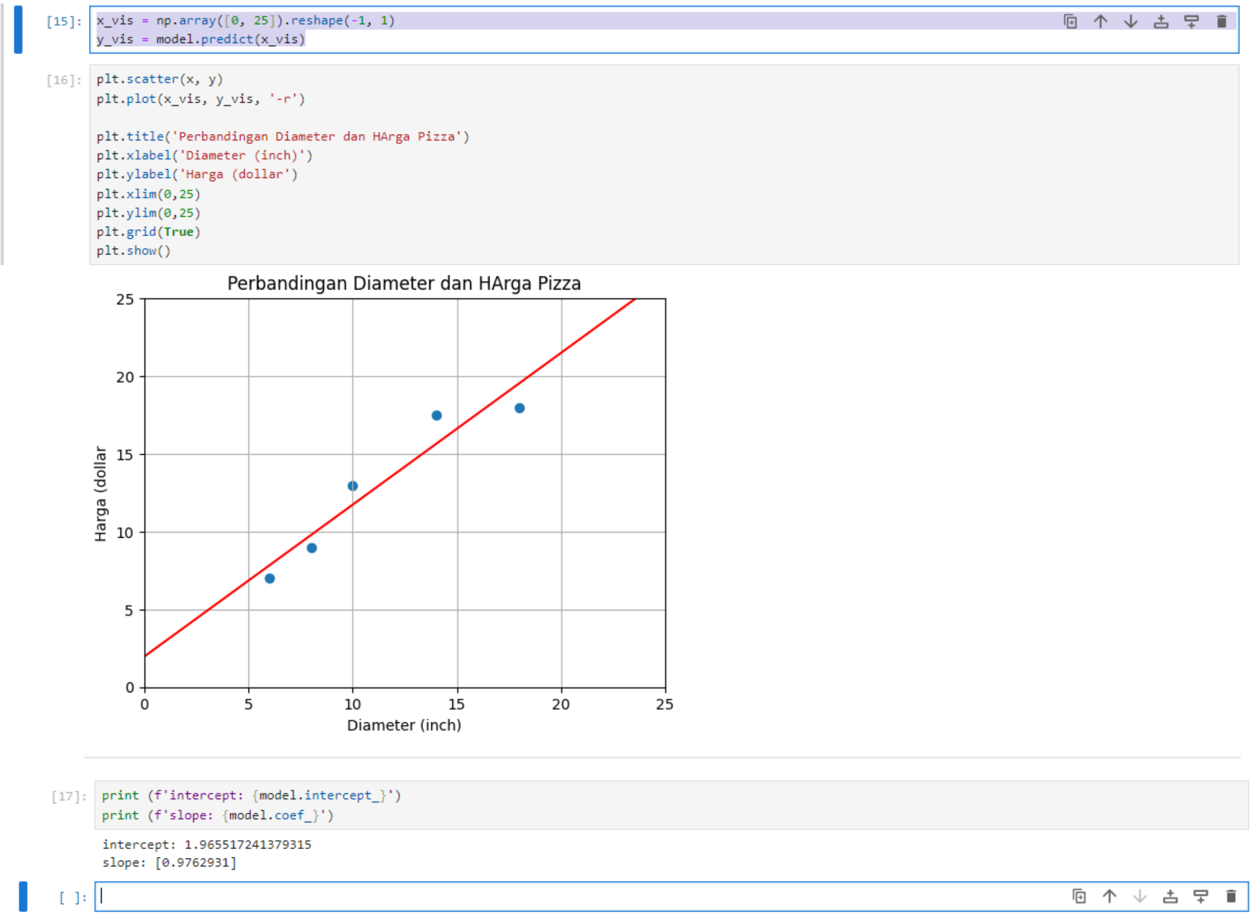
```
[ ]:
```

## Training Simple Linear Regression Model

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

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

[14]: ▾ LinearRegression
      LinearRegression()
```

```
[ ]:
```

## 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]:

```
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()
```

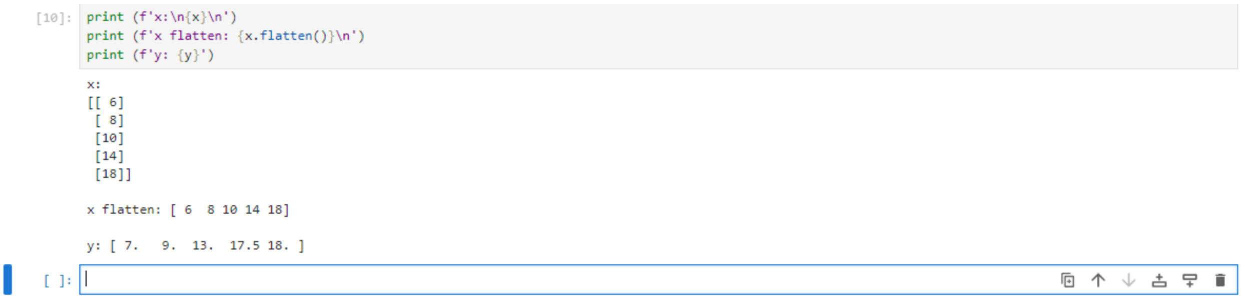
[17]:

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

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

[ ]:

## Kalkulasi nilai slope



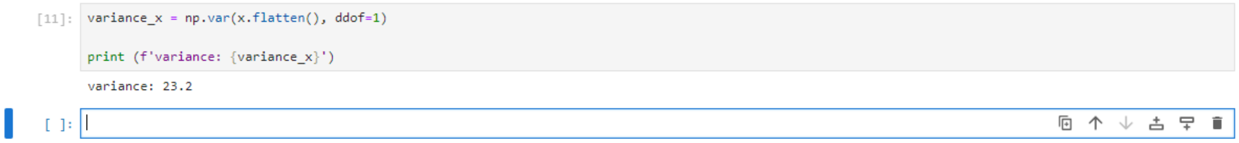
[10]:

```
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. ]
```

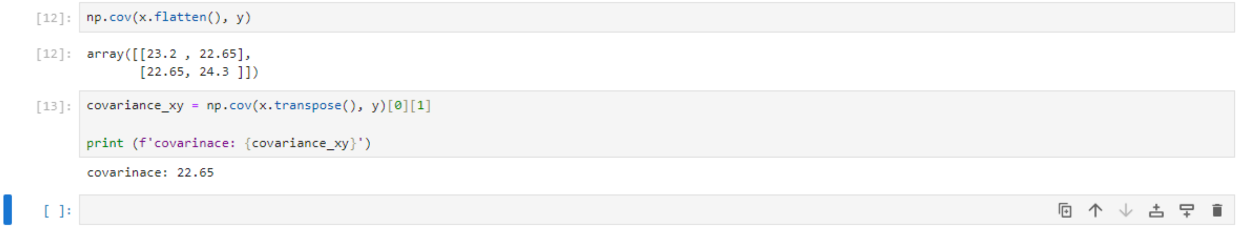
[ ]:

## Variance



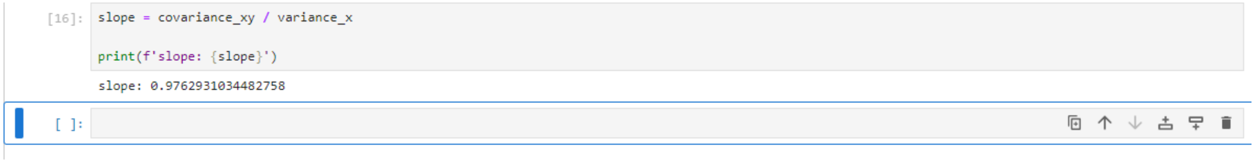
```
[11]: variance_x = np.var(x.flatten(), ddof=1)

      print (f'variance: {variance_x}')

      variance: 23.2

[ ]:
```

## Covariance



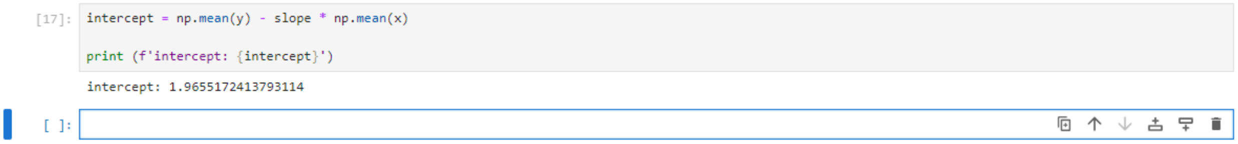
```
[12]: np.cov(x.flatten(), y)

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

[13]: covariance_xy = np.cov(x.transpose(), y)[0][1]

      print (f'covarinace: {covariance_xy}')

      covarinace: 22.65

[ ]:
```

## Slope



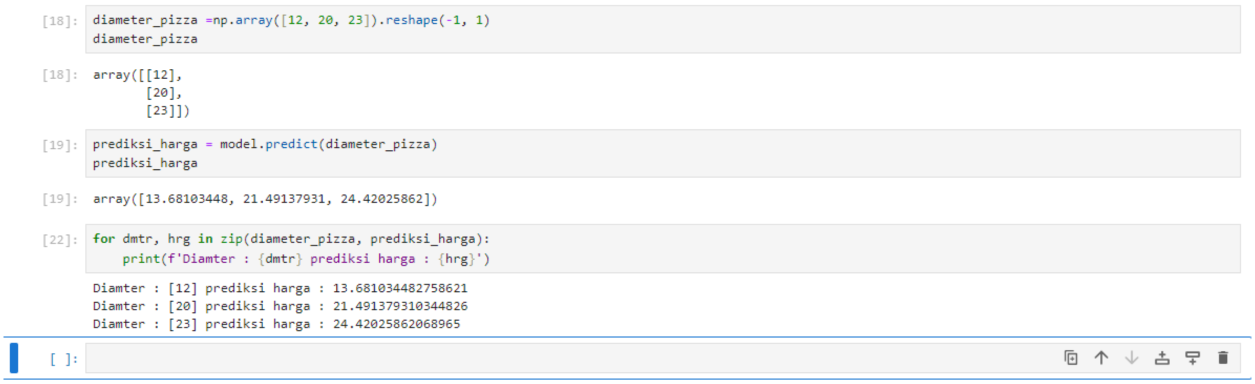
```
[16]: slope = covariance_xy / variance_x

      print(f'slope: {slope}')

      slope: 0.9762931034482758

[ ]:
```

## Kalkukasi nilai intercept



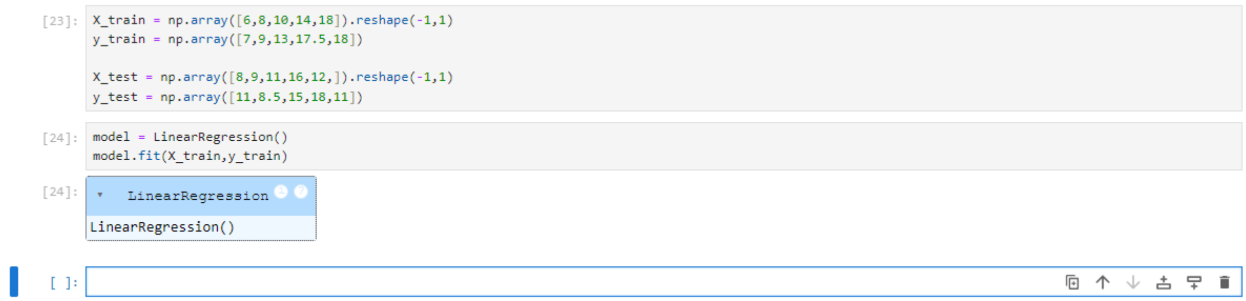
```
[17]: intercept = np.mean(y) - slope * np.mean(x)

      print (f'intercept: {intercept}')

      intercept: 1.9655172413793114

[ ]:
```

# Prediksi harga pizza dengan Simple Linear Regression Model

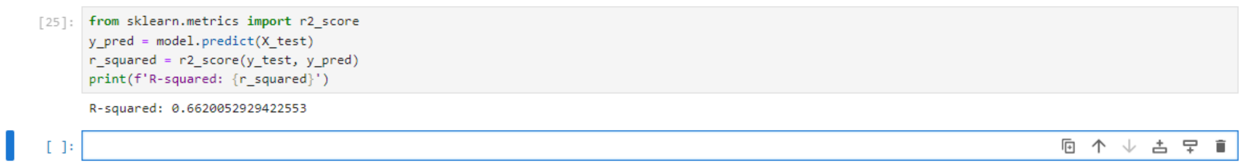


```
[18]: diameter_pizza =np.array([12, 20, 23]).reshape(-1, 1)
      diameter_pizza

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

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

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

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

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

[ ]:
```

# Evaluasi model dengan Coefficient of Determination | R Squared



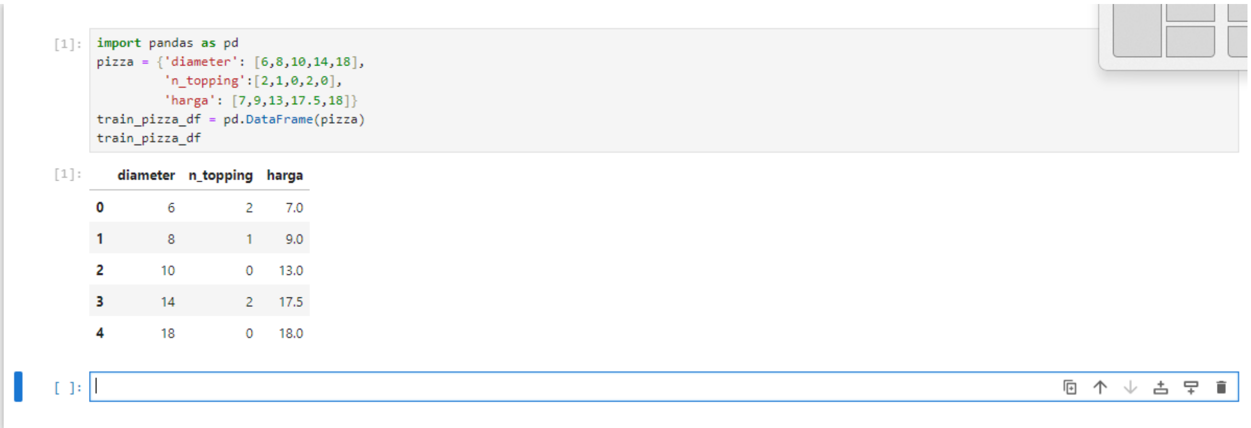
```
[23]: X_train = np.array([6,8,10,14,18]).reshape(-1,1)
      y_train = np.array([7,9,13,17.5,18])

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

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

[24]: ▾ LinearRegression
      LinearRegression()

[ ]:
```



```
[25]: 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

[ ]:
```

## Persiapan sample dataset

### Training Dataset



[1]:

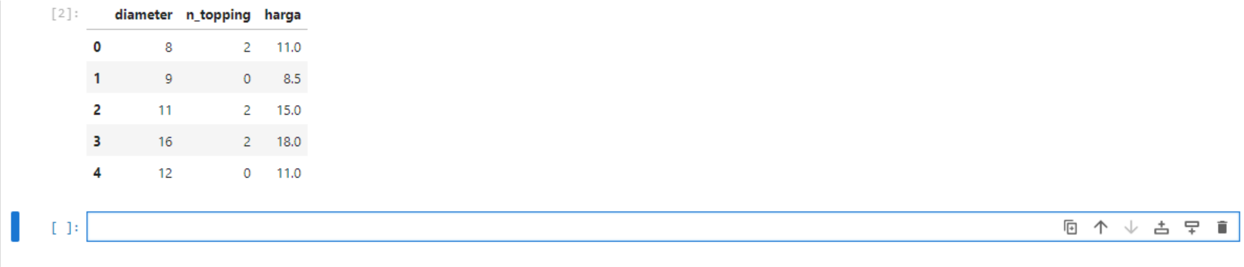
```python
import pandas as pd
pizza = {'diameter': [6,8,10,14,18],
         'n_topping':[2,1,0,2,0],
         'harga': [7,9,13,17.5,18]}
train_pizza_df = pd.DataFrame(pizza)
train_pizza_df
```

[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 |

[ ]:

### Testing Dataset

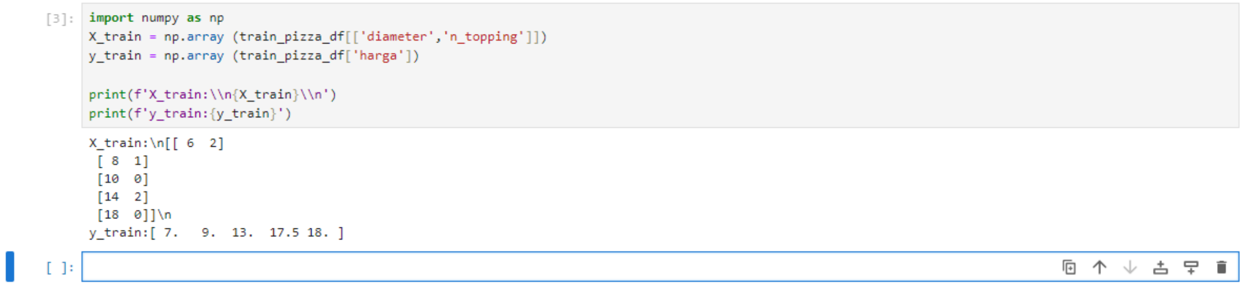


[2]:

|   | 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

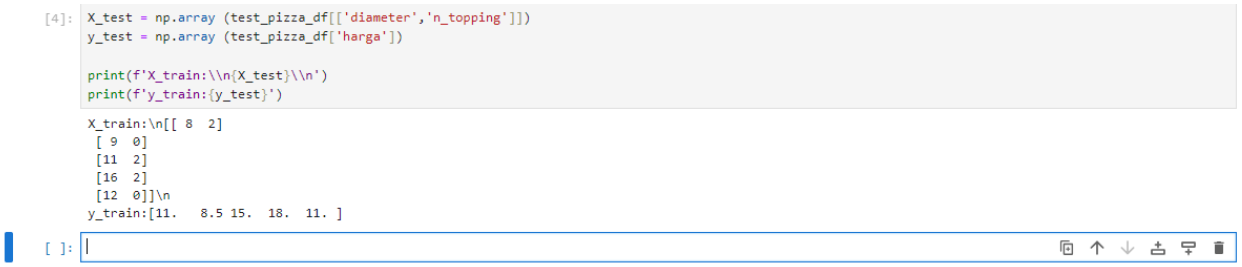


[3]:

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

[ ]:



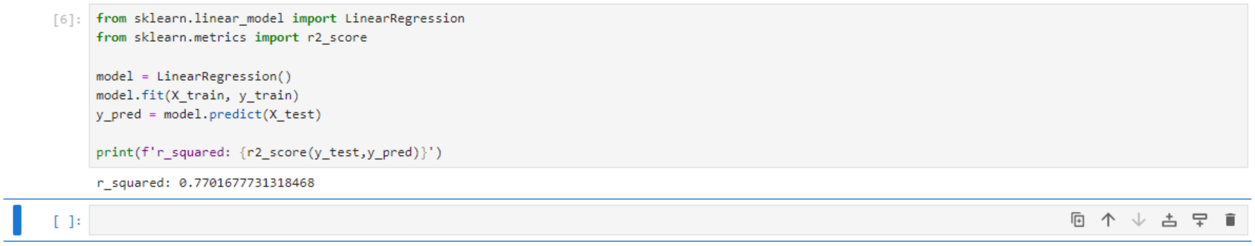
[4]:

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

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

```
X_train:\n[[ 8  2]
 [ 9  0]
 [11  2]
 [16  2]
 [12  0]]\n
y_train:[11.   8.5 15.  18.  11. ]
```

[ ]:

# Multiple Linear Regression



[6]:

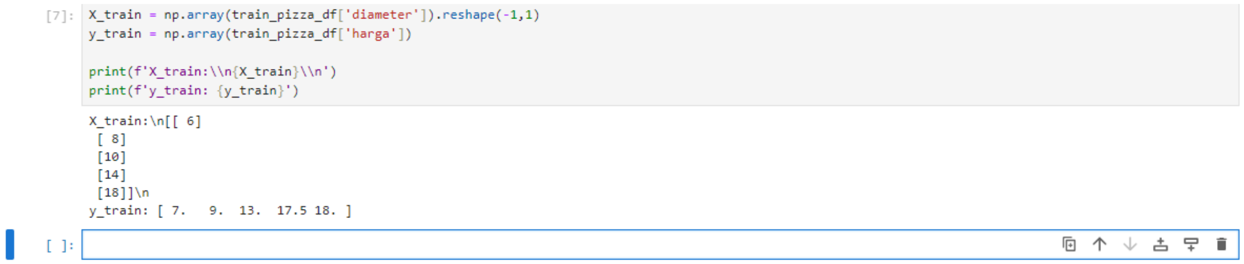
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

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



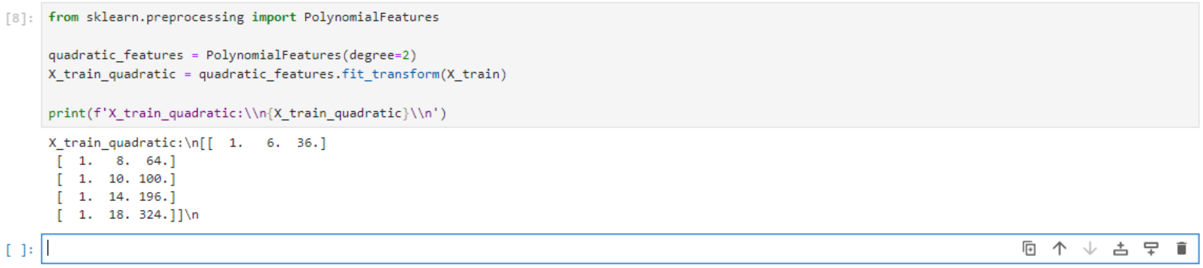
[7]:

```
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:\n[[ 6]
 [ 8]
 [10]
 [14]
 [18]]\n
y_train: [ 7.   9.  13.  17.5 18. ]
```

[ ]:

## Polynomial Features



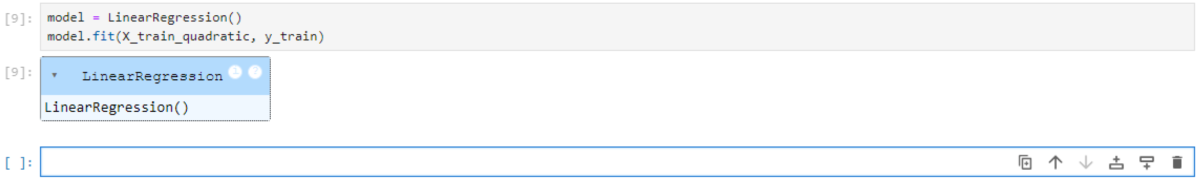
[8]:

```
from sklearn.preprocessing import PolynomialFeatures

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

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

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

[ ]:

## Training Model



[9]:

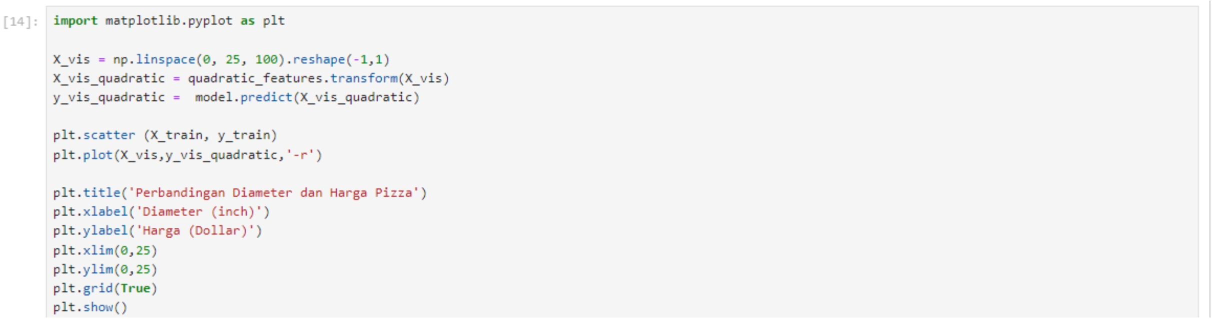
```
model = LinearRegression()
model.fit(X_train_quadratic, y_train)
```

[9]:

```
LinearRegression
LinearRegression()
```

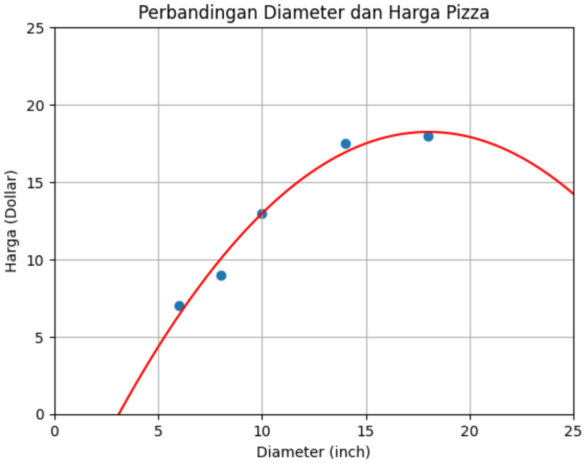
[ ]:

## Visualisasi Model



[14]:

```
import matplotlib.pyplot as plt

X_vis = np.linspace(0, 25, 100).reshape(-1,1)
X_vis_quadratic = quadratic_features.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()
```

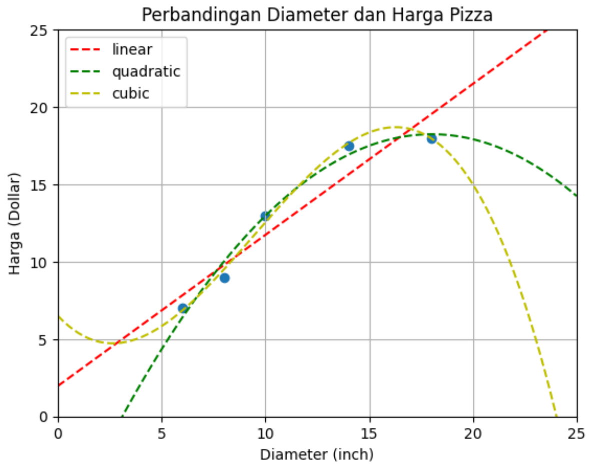


## Quadratic Polynomial Regression

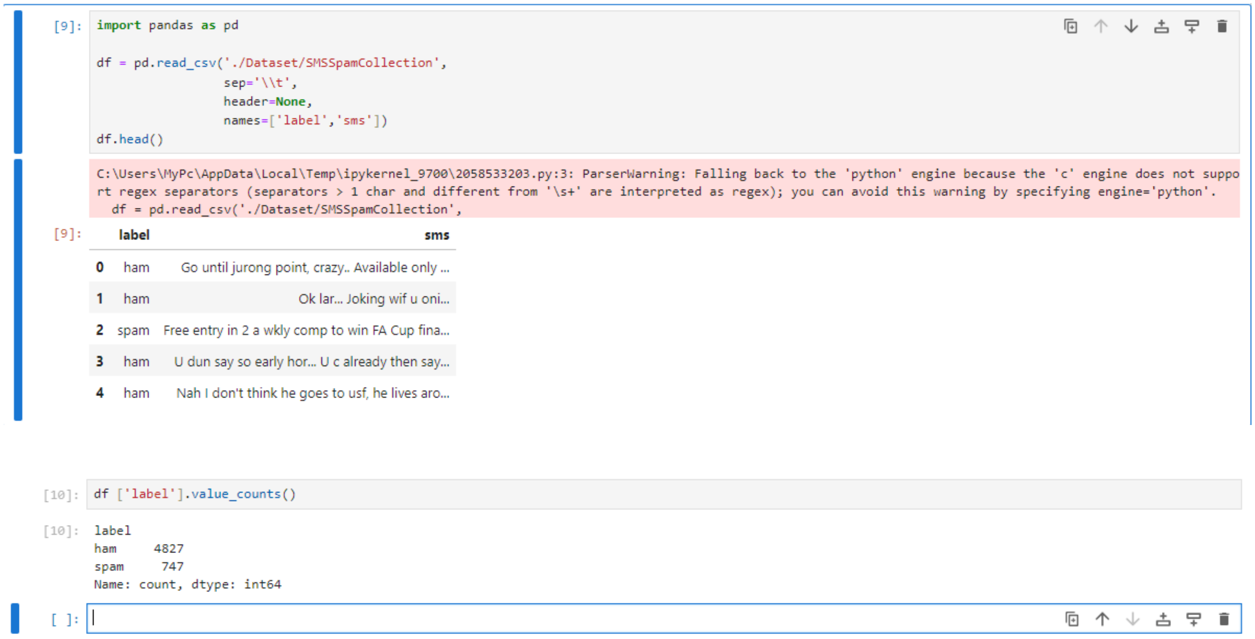
[16]:

```
# Training Set\n",
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()
```



## Dataset SMS Spam Collection Dataset



```
[9]: import pandas as pd

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

```
C:\Users\MyPc\AppData\Local\Temp\ipykernel_9700\2058533203.py:3: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not suppo
rt regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
  df = pd.read_csv('./Dataset/SMSSpamCollection',
```

[9]:

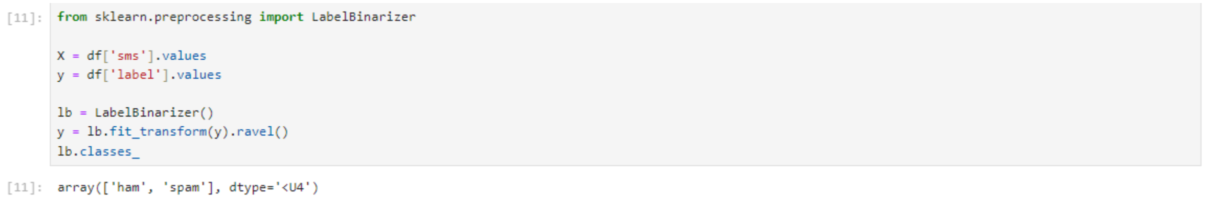
| | label | sms |
|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... |
| 1 | ham | Ok lar... Joking wif u oni... |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... |
| 3 | ham | U dun say so early hor... U c already then say... |
| 4 | ham | Nah I don't think he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

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

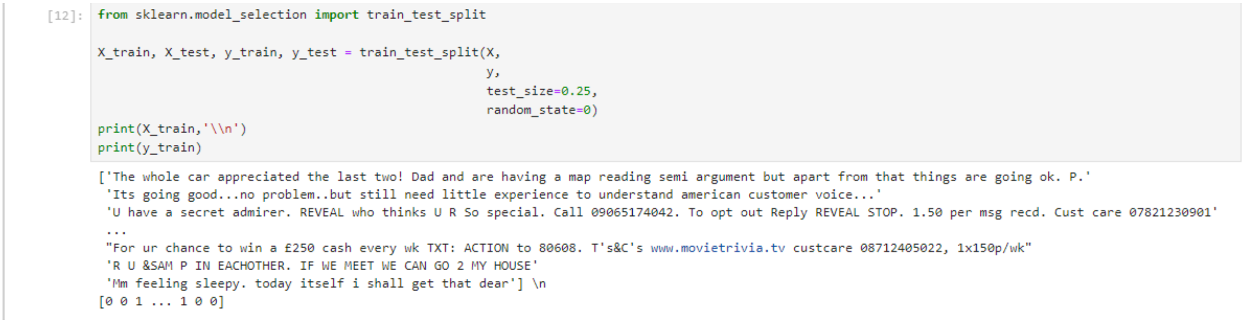
```
[10]: label
      ham     4827
      spam     747
      Name: count, dtype: int64
```

```
[ ]: |
```

## Training & Testing Dataset

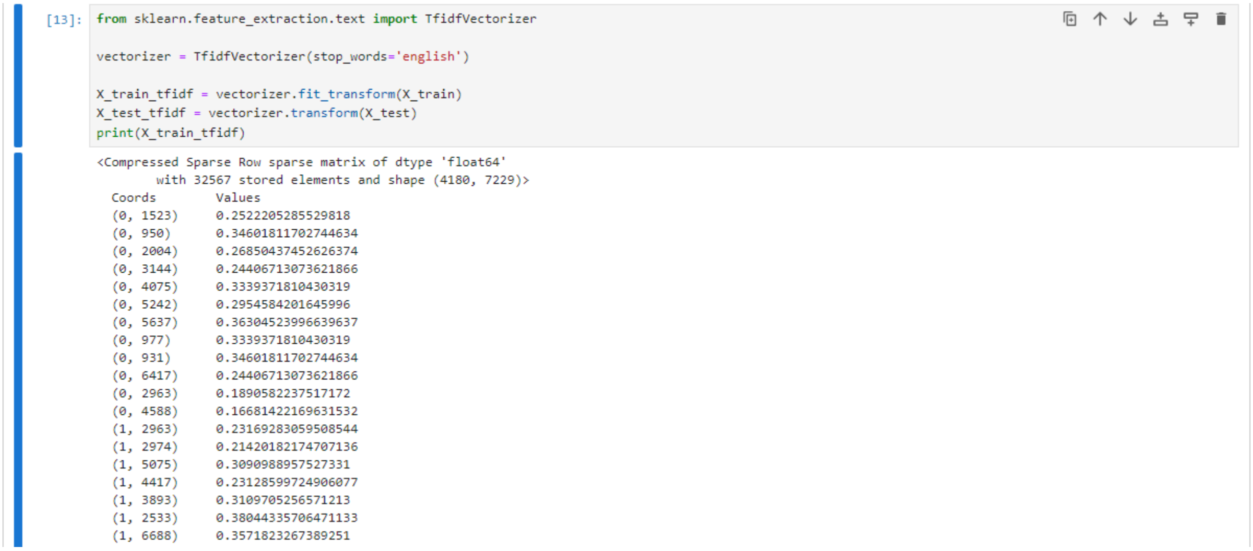


```
[11]: from sklearn.preprocessing import LabelBinarizer

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

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

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



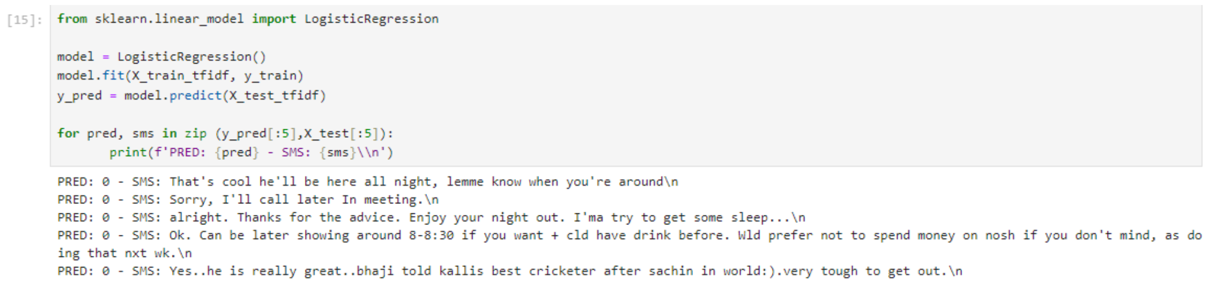
```
[12]: from sklearn.model_selection import train_test_split

      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)

      ['The whole car appreciated the last two! Dad and are having a map reading semi argument but apart from that things are going ok. P.'
       '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'
       ...
       "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'] \n
      [0 0 1 ... 1 0 0]
```

## Feature extraction dengan TF-IDF



```
[13]: 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 32567 stored elements and shape (4180, 7229)>
        Coords        Values
        (0, 1523)     0.2522205285529818
        (0, 950)      0.34601811702744634
        (0, 2004)     0.26850437452626374
        (0, 3144)     0.24406713073621866
        (0, 4075)     0.3339371810430319
        (0, 5242)     0.2954584201645996
        (0, 5637)     0.36304523996639637
        (0, 977)      0.3339371810430319
        (0, 931)      0.34601811702744634
        (0, 6417)     0.24406713073621866
        (0, 2963)     0.1890582237517172
        (0, 4588)     0.16681422169631532
        (1, 2963)     0.23169283059508544
        (1, 2974)     0.21420182174707136
        (1, 5075)     0.3090988957527331
        (1, 4417)     0.23128599724906077
        (1, 3893)     0.3109705256571213
        (1, 2533)     0.38044335706471133
        (1, 6688)     0.3571823267389251
```

## Binary Classification dengan Logistic Regression

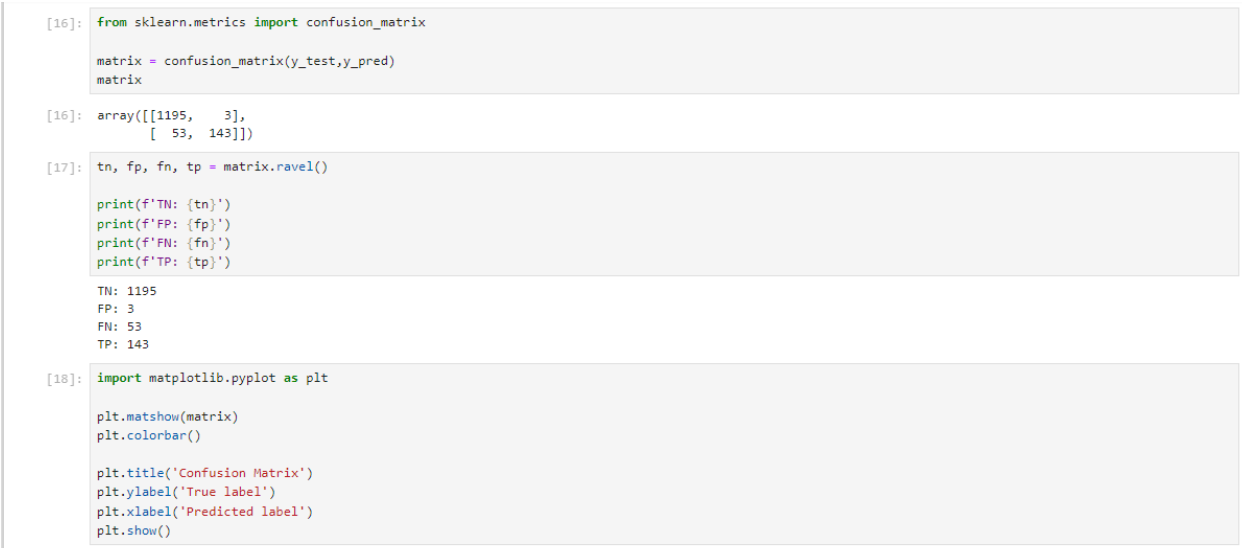


```
[15]: from sklearn.linear_model import LogisticRegression

      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: That's cool he'll be here all night, lemme know when you're around\n
PRED: 0 - SMS: Sorry, I'll call later In meeting.\n
PRED: 0 - SMS: alright. Thanks for the advice. Enjoy your night out. I'ma try to get some sleep...\n
PRED: 0 - SMS: Ok. Can be later showing around 8-8:30 if you want + cld have drink before. Wld prefer not to spend money on nosh if you don't mind, as do
ing that nxt wk.\n
PRED: 0 - SMS: Yes..he is really great..bhaji told kallis best cricketer after sachin in world:).very tough to get out.\n
```

## Evaluation Metrics pada Binary Classification Task

### Confusion matrix



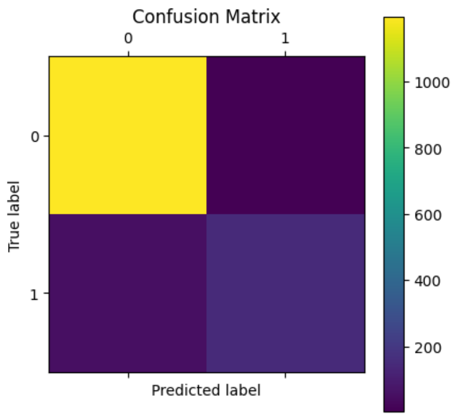
```
[16]: from sklearn.metrics import confusion_matrix

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

```
[16]: array([[1195,    3],
             [  53,  143]])
```

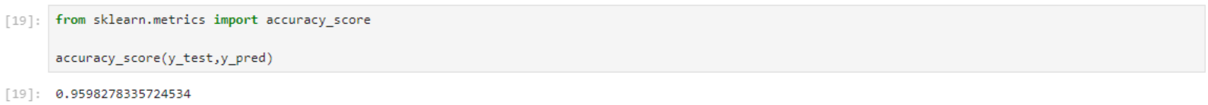
```
[17]: tn, fp, fn, tp = matrix.ravel()

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

```
TN: 1195
FP: 3
FN: 53
TP: 143
```

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

      plt.matshow(matrix)
      plt.colorbar()

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

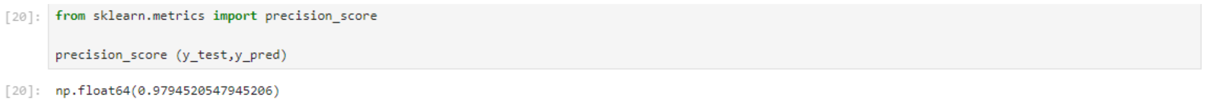


## Accuracy



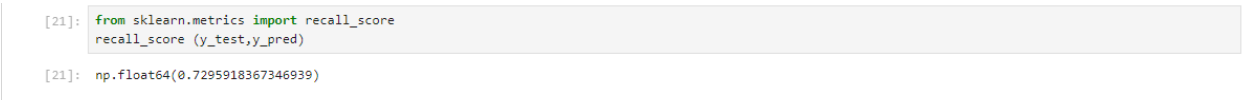
```
[19]: from sklearn.metrics import accuracy_score

      accuracy_score(y_test,y_pred)
[19]: 0.9598278335724534
```

## Precission & Recall



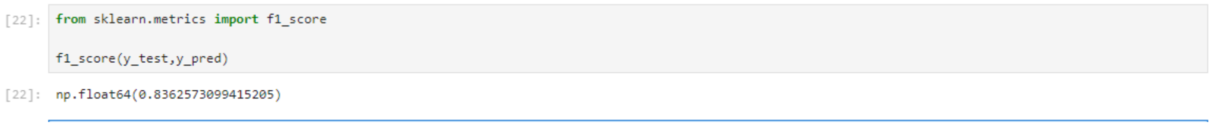
```
[20]: from sklearn.metrics import precision_score

      precision_score (y_test,y_pred)
[20]: np.float64(0.9794520547945206)
```

## Recall



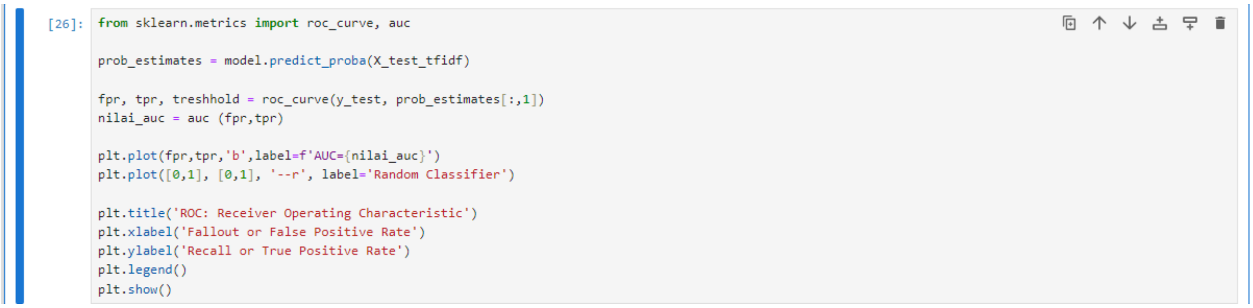
```
[21]: from sklearn.metrics import recall_score
      recall_score (y_test,y_pred)
[21]: np.float64(0.7295918367346939)
```

## F1-Score

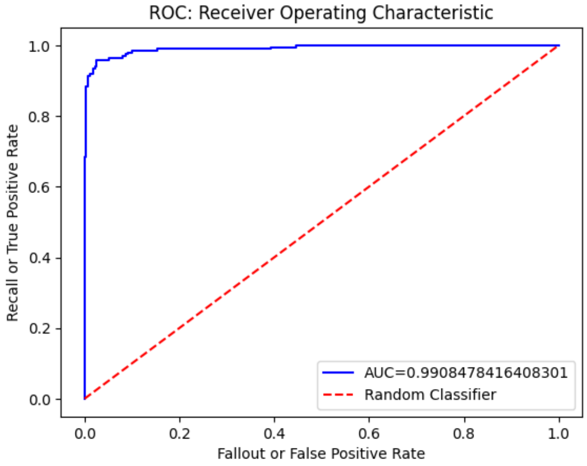


```
[22]: from sklearn.metrics import f1_score

      f1_score(y_test,y_pred)
[22]: np.float64(0.8362573099415205)
```

# ROC | Receiver Operating Characteristic



```
from sklearn.metrics import roc_curve, auc

prob_estimates = model.predict_proba(X_test_tfidf)

fpr, tpr, treshhold = 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=0.9908478416408301
Random Classifier