Simple Linear Regression

Muhammad Rofi Ariansyah

41155050210066

[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
```

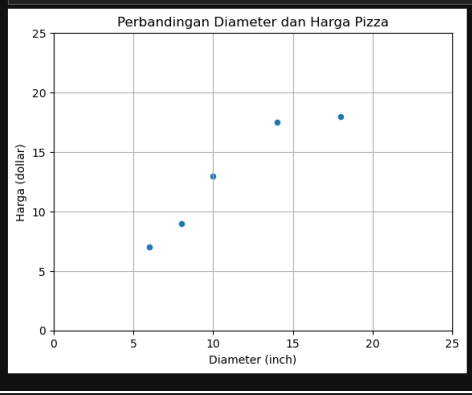
[1]:

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

Visualisasi Data

[25]:
```
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 Set

[3]:
```
# Penerapan simple linear Programming
# Penyesuaian Data set
import numpy as np

x = np.array(pizza_df['diameter']) # features
y = np. array(pizza_df['harga']) # target

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

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

[4]:
```
x = x.reshape(-1, 1)
x.shape
x
```

[4]:
```
array([[ 6],
       [ 8],
       [10],
       [14],
       [18]], dtype=int64)
```

Training Model Linier Regression Model ¶

[26]:
```
from sklearn.linear_model import LinearRegression

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

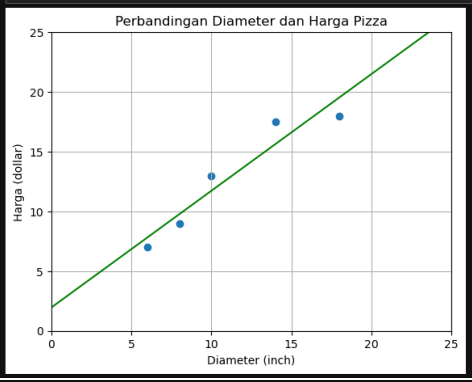
[26]:
```
LinearRegression
LinearRegression()
```

Visualisasi Simple Linear Regression Model

[27]:
```
x_vis = np.array([0,25]).reshape(-1,1)
y_vis =  model.predict(x_vis)
```

[7]:
```
plt.scatter(x, y)
plt.plot(x_vis, y_vis, '-g')

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()
```



Mencari Nilai Slope

[28]:
```
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

[29]:
```
variance_x = np.var(x.flatten(), ddof=1)
print(f'variance: {variance_x}')
```

```
variance: 23.2
```

Covariance

[30]:
```
np.cov(x.flatten(),y)
```

[30]:
```
array([[23.2 , 22.65],
       [22.65, 24.3 ]])
```

[12]:
```
covariance_xy = np.cov(x.flatten(),y)[0][1]
print(f'covariance: {covariance_xy}')
```

```
covariance: 22.65
```

Slope

[31]:
```
slope = covariance_xy / variance_x
print(f'slope: {slope}')
```

```
slope: 0.9762931034482758
```

Mencari Nilai Intercept

[32]:
```
intercept = np.mean(y)- slope * np.mean(x)
print(f'intercept: {intercept}')
```

```
intercept: 1.9655172413793114
```

Prediksi Harga Pizza

[33]:
```
diameter_pizza = np.array([12,20,23]).reshape(-1,1)
diameter_pizza
```

[33]:
```
array([[12],
       [20],
       [23]])
```

[16]:
```
prediksi_harga = model.predict(diameter_pizza)
prediksi_harga
```

[16]:
```
array([13.68103448, 21.49137931, 24.42025862])
```

[17]:
```
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
```

Training dan Testing Dataset

[34]:
```
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

[35]:
```
model = LinearRegression()
model.fit(X_train,y_train)
```

[35]:
```
LinearRegression()
```

[20]:
```
# Evaluasi Linear Regression Model dengan Cofficient of Determination atau R-Squared
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}')

# Nilai R-squred semakain mendekati 1 maka akan semakin baik bgtu pula sebaliknya
```

```
R-squared: 0.662005292942255 3
```

[21]:
```
# Mencari nilai R-squred
# SS_res
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
```

[22]:
```
# SS_tot
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
```

[23]:
```
# R-squared
r_squared  = 1- (ss_res / ss_tot)
print(f'R-squared : {r_squared}')
```

```
R-squared : 0.662005292942255 3
```

Multiple Linear Regression

Muhammad Rofi Ariansyah

41155050210066

[1]:
```
# Training Dataset
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

[3]:
```
# Testing Dataset
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

Preprocessing Dataset

[4]:
```
# Preprocessing Dataset
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]:
```
import numpy as np
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:
[[ 8  2]
 [ 9  0]
 [11  2]
 [16  2]
 [12  0]]

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

Penerapan Multiple Linear Regression

- Multiple Linear Regression merupakan generalisasi dari simple Linear Regression yang memungkinkan untuk menggunakan beberapa explanotory variables
- pada Multiple linear regression menggunakan lebih dari satu features untuk melakukan prediksi

[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

Polinomial Regression memodelkan hubungan antara independent variable X (features) dan dependent variable y(target) sebagai derajat dalam x

[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

[8]:
```
# Polinomial Features
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:
[[  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()
```

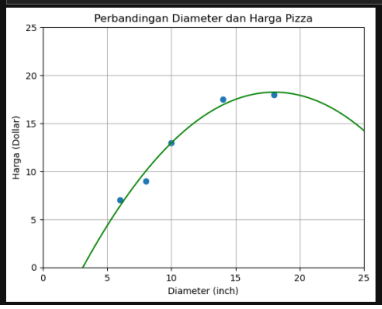
Visialisasi Model

[10]:
```
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,'-g')

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()
```



Polinomila Regression : Quadratic vs Cubic

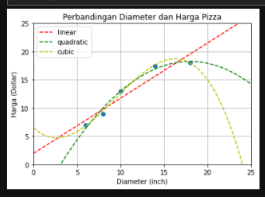
[14]:
```
# Training Set
plt.scatter (X_train, y_train) # plotting dataset

# 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()
```



Logistik Regression pada Binary Classification

Muhammad Rofi Ariansyah

41155050210066

[3]:
```
# Dataset : SMS Spam Collection Data set
import pandas as pd
df = pd.read_csv('file:///D:/MSIB/Jupyter/Machine Learning/Dataset/SMSSpamCollection',
                 sep='\t',
                 header=None,
                 names=['label','sms'])
df.head()
```

[3]:

	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...

[3]:
```
df ['label'].value_counts()
# kondisi pada output value_counts unbalance
```

[3]:
```
ham     4825
spam     747
Name: label, dtype: int64
```

[4]:
```
from sklearn.preprocessing import LabelBinarizer
X = df['sms'].values # features
y = df['label'].values # Target

lb = LabelBinarizer()
y = lb.fit_transform(y).ravel()
lb.classes_
# pada aouput ham di ibaratkan berkolerasi pada  classes 0(ham) dan spam(1) di ibaratkan classes 2
```

[4]:
```
array(['ham', 'spam'], dtype='<U4')
```

[5]:
```
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, # porsi untuk testing data set adalah 25% dan 75% akan digunakan sebagai training
                                                    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 0782123090
1'
 '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]
```

[6]:
```
# Features Extraction Dengan TF-IDF
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)
```

```
  (0, 6903)     0.3591386422223876
  (0, 2006)     0.2898082580285881
  (0, 900)      0.4114867709157148
  (0, 6739)     0.3546359942830148
  (0, 2554)     0.3825278811525034
  (0, 3926)     0.3126721340000456
  (0, 4453)     0.2297719954323795
  (0, 5123)     0.308974289326673
  (0, 3007)     0.21421364306658514
  (0, 2997)     0.23173982975834367
  (1, 36)       0.28902673040368515
  (1, 1548)     0.18167737976542422
  (1, 2003)     0.2711077935907125
  (1, 5301)     0.2711077935907125
  (1, 4358)     0.17341410292348694
  (1, 532)      0.20186022353306565
  (1, 6131)     0.16142609035094446
  (1, 5394)     0.16464655071448758
  (1, 4677)     0.24039776602646504
  (1, 216)      0.28902673040368515
  (1, 6013)     0.20089911182610476
  (1, 6472)     0.24039776602646504
  (1, 5441)     0.5009783758205715
  (1, 799)      0.25048918791028574
  (1, 5642)     0.24344998442301355
```


[7]:
```
# Binary Classification dengan Logistic Regression
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train_tfidf, y_train) # training dengan memanggil method fit
y_pred = model.predict(X_test_tfidf) # melakukan  prediksi

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 knw who is dat girl?? "Margaret Hello" She is d gi
rlfrnd f Grahmbell who invnted telphone... . . . . Moral:One 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..
```

Evaluation Mertics pada Binary Classification

- Confusion Matrix
- Accuracy
- Precission & Recall
- F1 Score
- ROC

Terminologi Dasar

- True Positive (TP) >> keduanya merepresentasikan hasil prediksi atau klasifikasi yang benar
- True Negative (TN) >> sesuatu yang bernilai negatif telah dengan tepat diprediksi sebagai negatif oleh model
- False Positive (FP) >> keduanya merepsentasikan hasil prediksi atau klasifikasi yang salah atau sesuatu yang bernilai negatif telah keliru di prediksi sebagai positif oleh model
- False Negative (FN) >> sesuatu yang bernilai positif telah keliru di prediksi sebagai negatif oleh model

Confusion Matrix

Confusion matrix sering juga dikenal sebagai error matrix berberan untuk menampilkan nilai TP,TN,FP,FN.

[8]:
```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test,y_pred)
matrix
```

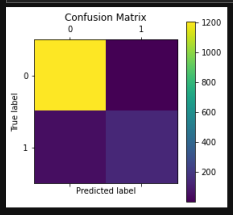
[8]:
```
array([[1207,    1],
       [  48,  137]], dtype=int64)
```

[9]:
```
tn, fp, fn, tp = matrix.ravel() # menganti matrix ke dalam array satu dimensi
print(f'TN: {tn}')
print(f'FP: {fp}')
print(f'FN: {fn}')
print(f'TP: {tp}')
```

```
TN: 1207
FP: 1
FN: 48
TP: 137
```

[11]:
```
# Visualize Data
import matplotlib.pyplot as plt
plt.matshow(matrix)
plt.colorbar()

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



Nilai 0 pada sumbu y (True label) merepresentasikan nilai yang sebenarnya dan nilai 0 pada sumbu x (Predicted label) yang artinya pada sumbu y nilai yang sebanernya adalah noldan prediksinya adalah nol.

Accuracy

Accurasi mengukur porsi dari hasil prediksi yang tepat

[12]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

[12]:
```
0.964824120603015
```

Precission & Recall

Selain menggunakan Accuracy, performa dari suatu classifier umumnya juga diukur berdasarkan nilai Precission & Recall.

[13]:
```
# Preccission or Positive Predictive Value (PPV)
from sklearn.metrics import precision_score
precision_score (y_test,y_pred)
```

[13]:
```
0.9927536231884058
```

[14]:
```
# Recall or True Positive Rate (TPR) or Sensitivity
from sklearn.metrics import recall_score
recall_score (y_test,y_pred)
```

[14]:
```
0.7405405405405405
```

F1-Score

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

[15]:
```
# F1-Score
from sklearn.metrics import f1_score
f1_score(y_test,y_pred)
```

[15]:
```
0.8482972136222909
```

ROC (Receiver Operating Characteristic)

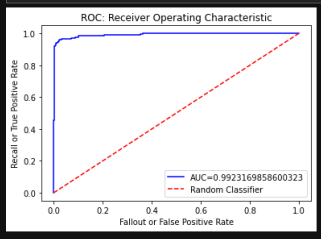
ROC menawarkan visualisasi terhadap performa dari classifier dengan membandingkan nilai Recall (TPR) dan Nilai Fallout (FPR)

[16]:
```
# 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()
```



kerika nilai kurva ROC lebih condong mengarah kearah atas kanan makan model yang dibuat semakin baik dan apabila kurva ROC lebih condong ke arah kanan bawah makan model semakin buruk.