Laporan Tugas 2 Machine Learning

Dosen : Estiyan Dwepriyoko, S.kom., M.T

Disusun Oleh :

Rio Putra Anugrah
41155050210044
TIF – A2

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS LANGLANGBUANA
BANDUNG
2023/2024

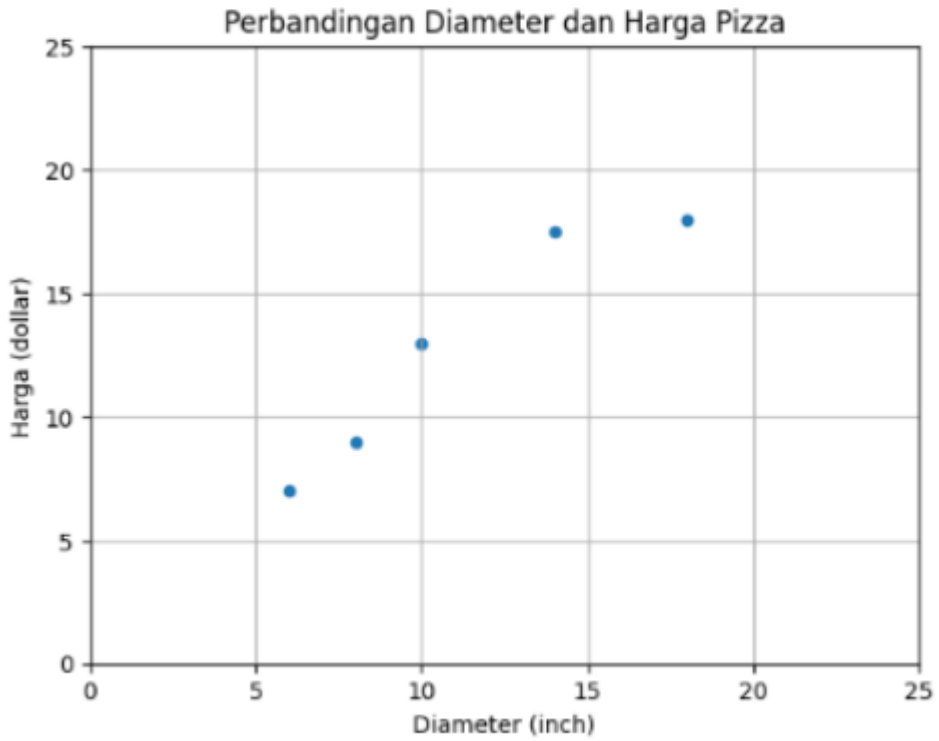
1.1 Sample dataset

```
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
```

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

1.2 Visualisasi dataset

```
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()
```



1.3 Transformasi dataset

```
]:  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. ]
```

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

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

```
]:  X
```

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

1.4 Training Simple Linear Regression Model

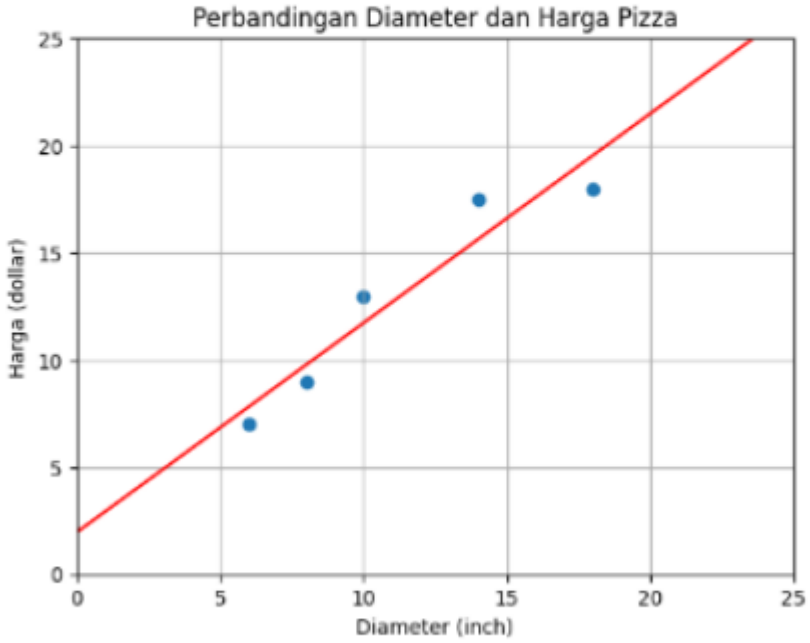
```
]:  from sklearn.linear_model import LinearRegression

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

```
]:  LinearRegression
    LinearRegression()
```

1.5 Visualisasi Simple Linear Regression Model | Penjelasan persamaan garis linear

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

```
[8]: 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()
```




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

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

1.6 Kalkulasi nilai slope

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

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

```
variance: 23.2
```

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

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

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

    print(f'covariance: {covariance_xy}')

    covariance: 22.650000000000002

6]: slope = covariance_xy / variance_x

    print(f'slope: {slope}')

    slope: 0.976293103448276

1.
```

1.7 Kalkukasi nilai intercept

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

   print(f'intercept: {intercept}')

   intercept: 1.9655172413793096

]:
```

1.8 Prediksi harga pizza dengan Simple Linear Regression Model

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

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

9]: prediksi_harga = model.predict(diameter_pizza)
    prediksi_harga

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

9]: 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

]:
```

1.9 Evaluasi model dengan Coefficient of Determination | 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}')
```

```
R-squared: 0.6620052929422553
```

1.10 Kalkulasi nilai R Squared | Coefficient of Determination

```
4]: 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
```

```
5]: 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
```

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

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

```
R-squared: 0.6620052929422553
```

```
]:
```

2.1 Persiapan sample 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
```

	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

```
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
```

	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

2.2 Preprocessing dataset

File Edit View Run Kernel Settings Help

Code

```
[30]: 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. ]
```

```
[31]: 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:\n {y_test}')
```

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

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

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

```
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
```

2.4 Pengenalan Polynomial Regression | Apa itu Polynomial Regression?

```
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.]]
```

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

```
LinearRegression
LinearRegression()
```

2.5 Quadratic Polynomial Regressio

```
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()
```

2.6 Linear Regression vs Quadratic Polynomial Regression vs Cubic Polynomial Regression

```
# 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()
```

3.3 Pembagian training dan testing set

Code JupyterLab Python 3 (ipykernel)

```
from sklearn.preprocessing import LabelBinarizer

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

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

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

```
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)
```

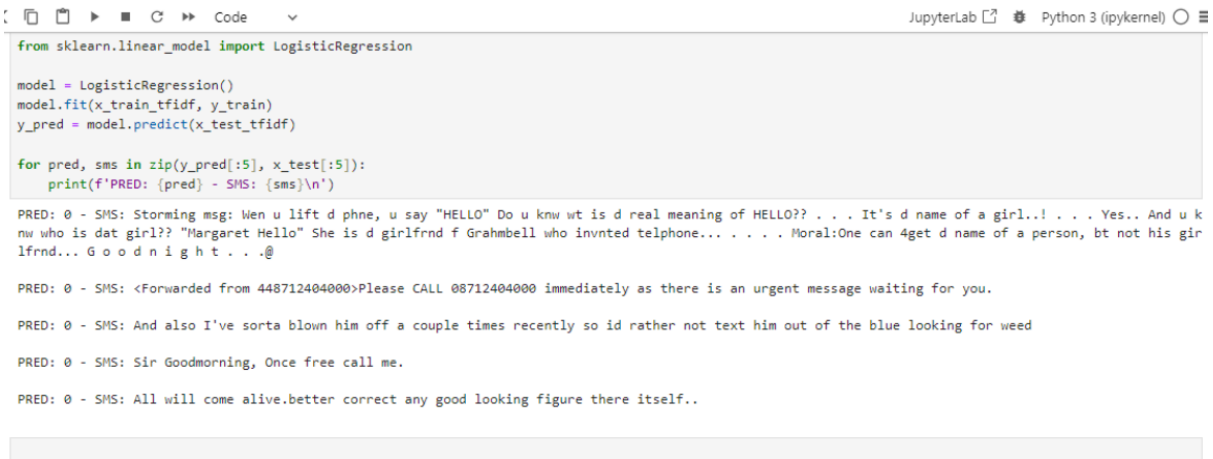
```
['Its going good...no problem..but still need little experience to understand american customer voice...'
 'U have a secret admirer. REVEAL who thinks U R So special. Call 09065174042. To opt out Reply REVEAL STOP. 1.50 per msg recd. Cust care 07821230901'
 'Ok...' ...
 "For ur chance to win a £250 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]
```

3.4 Feature 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)
```

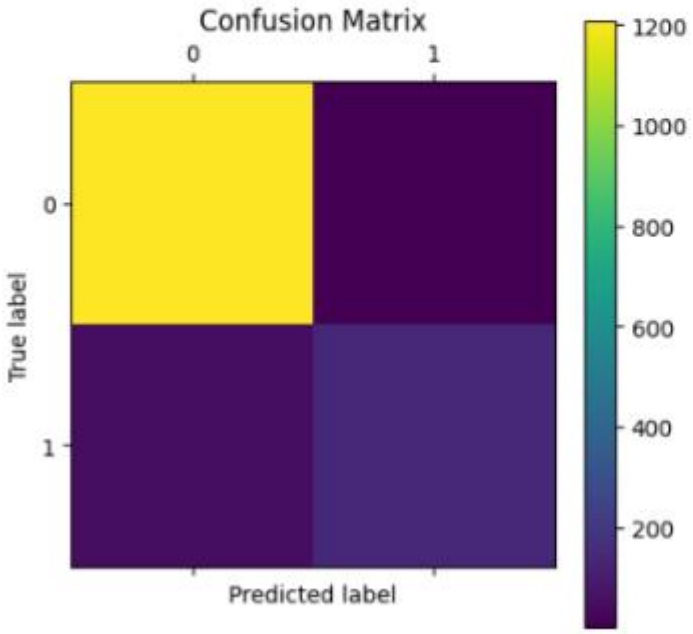
```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 32656 stored elements and shape (4179, 7287)>
  Coords        Values
  (0, 2997)     0.23173982975834367
  (0, 3007)     0.21421364306658514
  (0, 5123)     0.308974289326673
  (0, 4453)     0.2297719954323795
  (0, 3926)     0.3126721340000456
  (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
```

3.5 Binary Classification dengan Logistic Regression



3.7 Pengenalan Confusion Matrix

```
]: import matplotlib.pyplot as plt

   plt.matshow(matrix)
   plt.colorbar()

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



3.9 Pengenalan Precision dan Recall

```
]: from sklearn.metrics import recall_score
   recall_score(y_test, y_pred)
```

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

```
]:
```

3.10 Pengenalan F1 Score | F1 Measure

```
]: from sklearn.metrics import f1_score

   f1_score(y_test, y_pred)
```

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

```
]:
```

3.11 Pengenalan ROC | Receiver Operating Characteristic

```
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()
```