Nama : Miladyna Fauzia

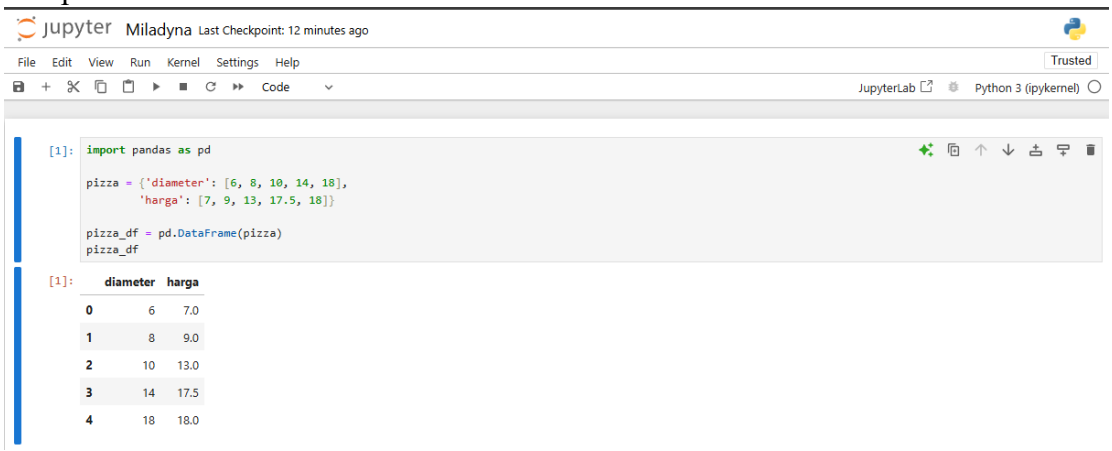
NPM : 41155050210023

Kelas : INF A1

Pertemuan 2

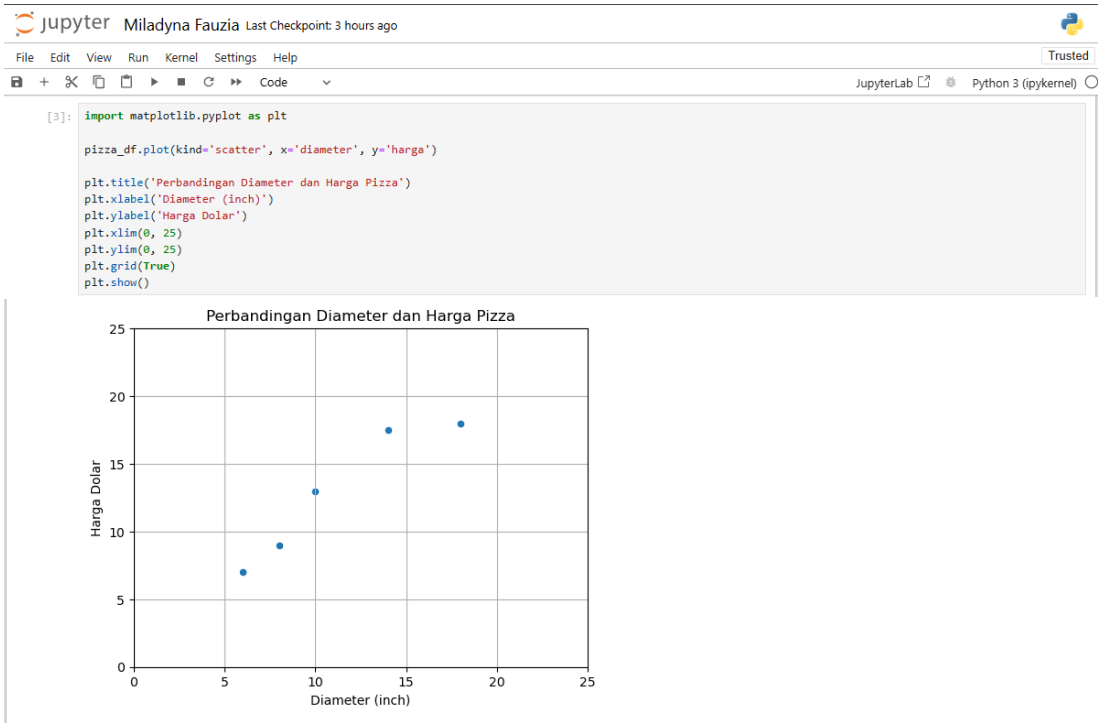
1. Lakukan praktek dari https://youtu.be/lcjq7-2zMSA?si=f4jWJR6lY8y0BZKl dan buat screen shot hasil run dengan nama anda pada hasil run tersebut. Praktek tersebut yaitu

1.1. Sample dataset



	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



1.3. Transformasi dataset

jupyter Miladyna Fauzia Last Checkpoint: 3 hours ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

```
[5]: 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. ]
```

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

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

```
[11]: X
```

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

1.4. Training Simple Linear Regression Model

jupyter Miladyna Fauzia Last Checkpoint: 3 hours ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

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

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

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

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

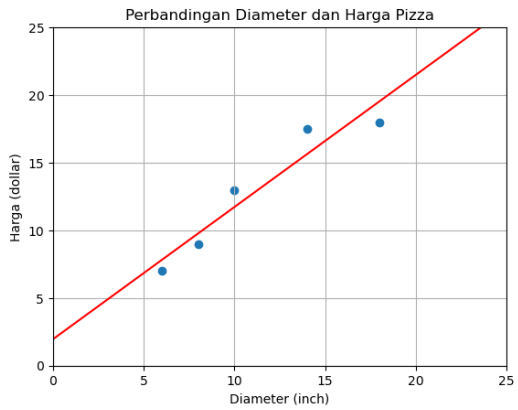
jupyter Miladyna Fauzia Last Checkpoint: 3 hours ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

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

```
[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 $y = \alpha + \beta x$

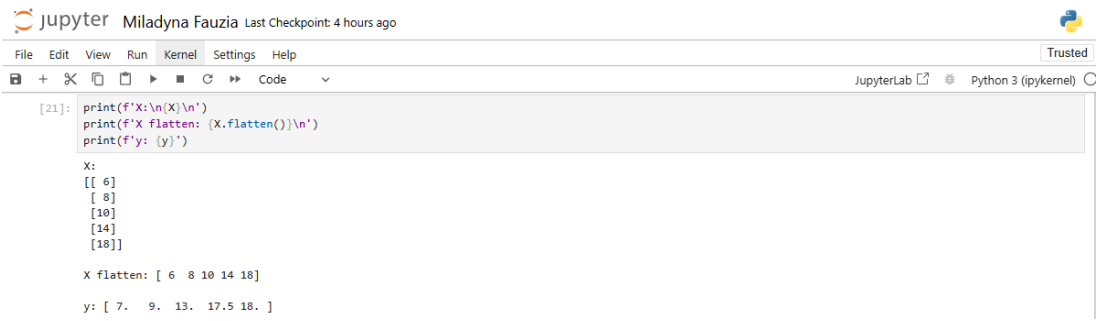
- y: response variable
- x: explanatory variable
- α: intercept
- β: slope

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

      intercept: 1.965517241379315
      slope:[0.9762931]
```

1.6. Mencari nilai slope

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

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



Variance

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

      print(f'variance: {variance_x}')

      variance: 23.2
```

Covariance

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

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

[27]: covariance_xy = np.cov(X.transpose(), y)[0][1]

      print(f'covariance: {covariance_xy}')

      covariance: 22.65
```

Slope

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

      print(f'slope: {slope}')

      slope: 0.9762931034482758
```

1.7. Mencari nilai intercept

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

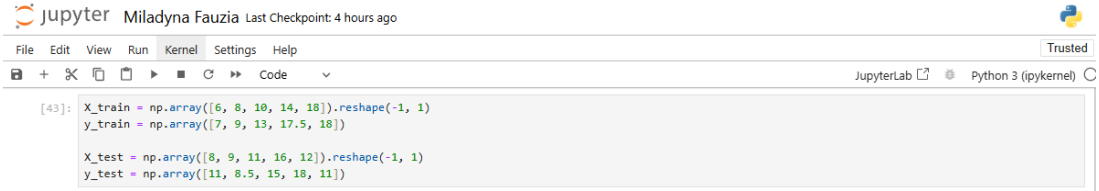
$$\alpha = y - \beta x$$

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

      print(f'intercept: {intercept}')

      intercept: 1.9655172413793114
```

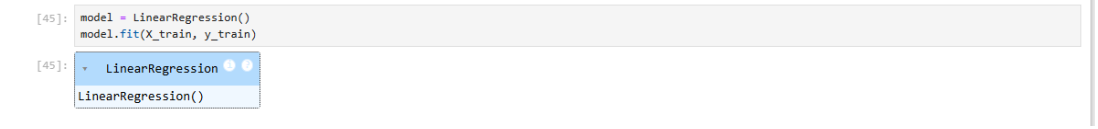
1.8. Prediksi harga pizza dengan Simple Linear Regression Model

1.9. Evaluasi Simpel Linear Regression Model

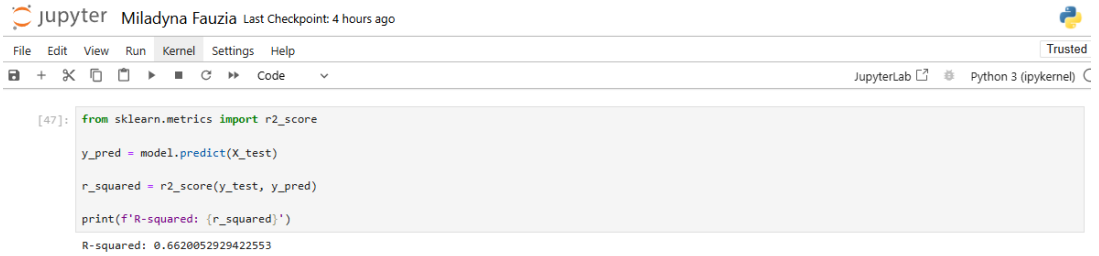
Training dan Testing Dataset



Training Simpel Linear Regression Model



1.10. Evaluasi model dengan Coefficient of Determination | R Squared (R^2)



1.11. Kalkulasi nilai R Squared | Coefficient of Determination

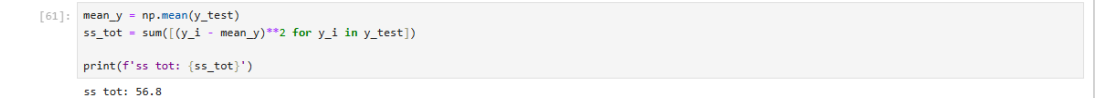
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

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

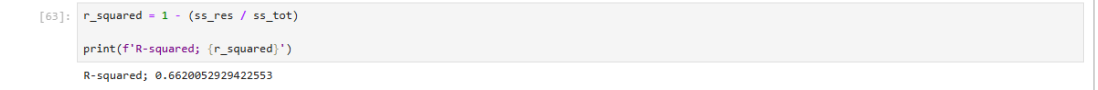
$$SS_{tot} = \sum_{i=1}^{n} (y_i - f(y)^2$$

SS_{res}

SS_{tot}



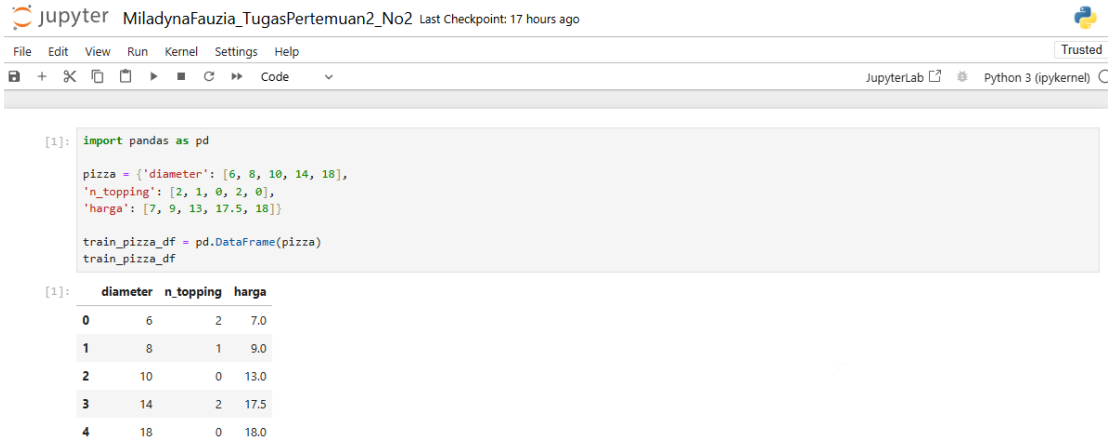
R^2



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. Sample dataset

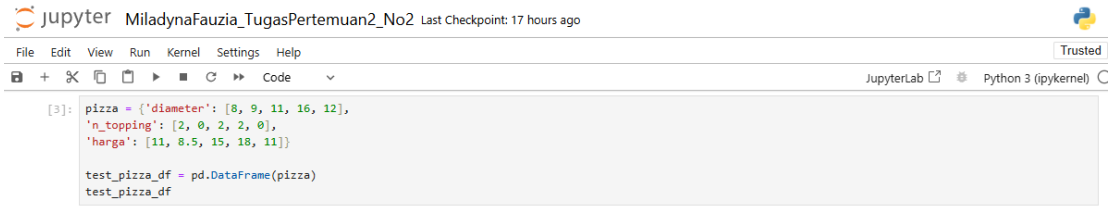
Training Dataset



[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



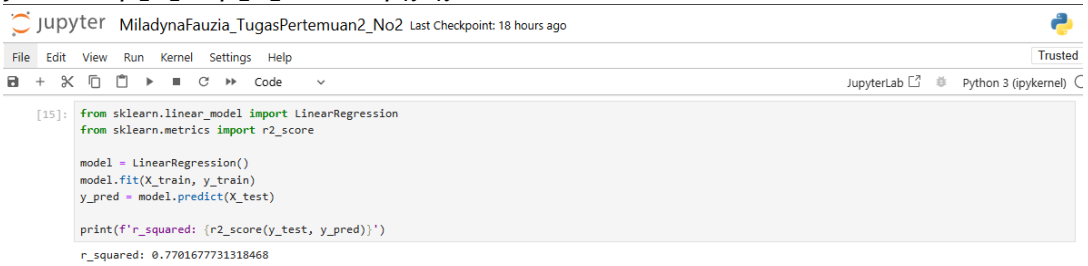
[3]:

	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

2.2. Preprocessing dataset

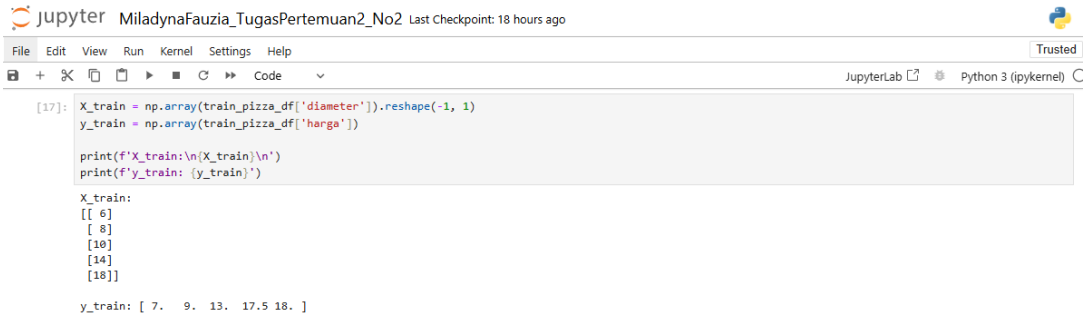
2.3. Pengenalan Multiple Linear Regression | Apa itu Multiple Linear Regression?
Multiple Linear Regression merupakan generalisasi dari Simple Linear Regression yang memungkinkan untuk menggunakan beberapa explanatory variables.

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



2.4. Pengenalan Polynomial Regression | Apa itu Polynomial Regression?
itu Polynomial Regression memodelkan hubungna antara independent variable x dan dependent variable y sebagai polynomial dalam x

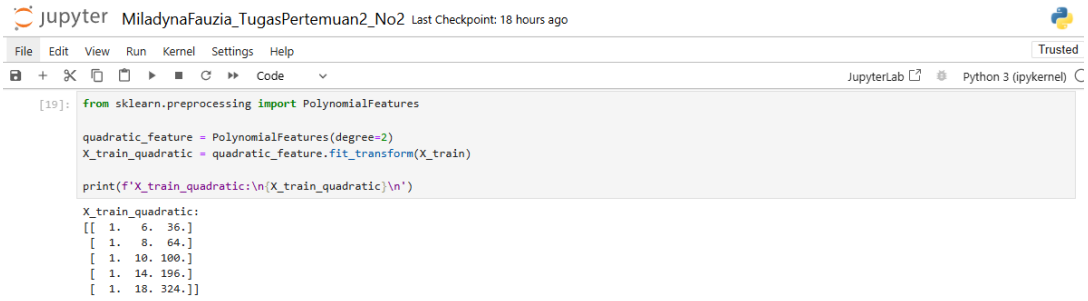
Proses Dataset



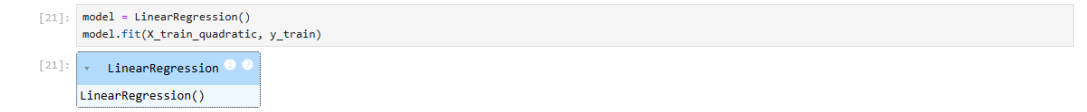
2.5. Quadratic Polynomial Regression

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

Polynomial Features



Training Model



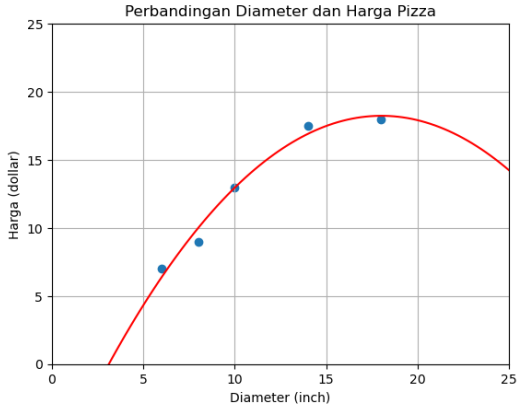
Visualisasi Model

jupyter MiladynaFauzia_TugasPertemuan2_No2 Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[23]:
```
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()
```



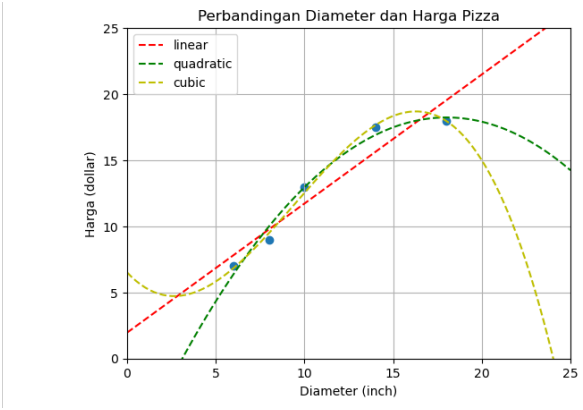
2.6. Linear Regression vs Quadratic Polynomial Regression vs Cubic Polynomial Regression

jupyter MiladynaFauzia_TugasPertemuan2_No2 Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[32]:
```
#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. 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: Logistic Regression pada Binary Classification Task

3.1. Formula dasar pembentuk Logistic Regression | Fungsi Sigmoid

Simple Linear Regression

- $y = a + \beta x$
- $g(x) = a + \beta x$

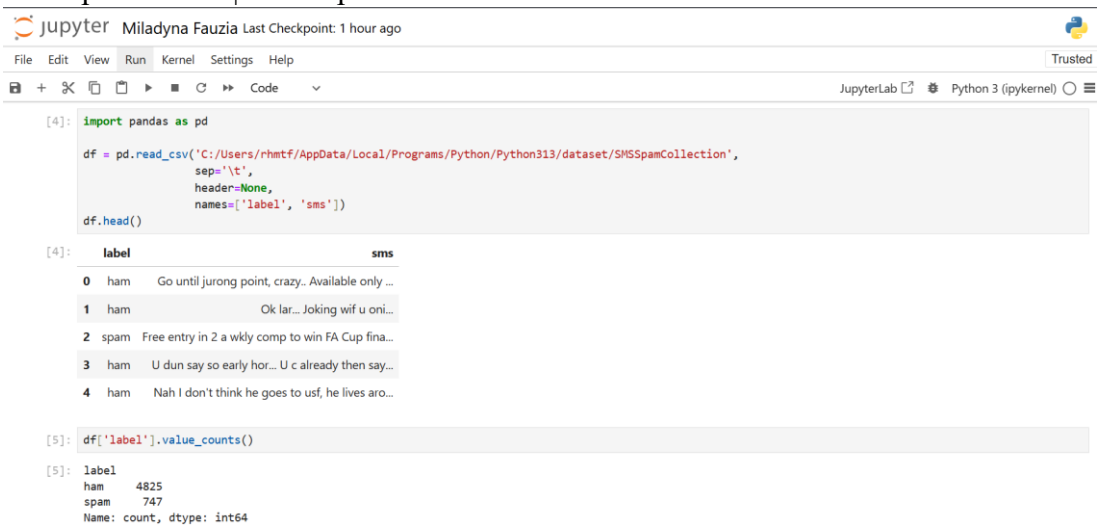
Multiple Linear Regression

- $y = a + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$
- $g(X) = a + \beta X$

Logistic Regression

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

3.2. Persiapan dataset | SMS Spam Collection Dataset



3.3. Pembagian training dan testing dataset

jupyter Miladyna Fauzia Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[7]:
```
from sklearn.preprocessing import LabelBinarizer

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

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

[7]:
```
array(['ham', 'spam'], dtype='<U4')
```

[9]:
```
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

jupyter Miladyna Fauzia Last Checkpoint: 1 hour ago

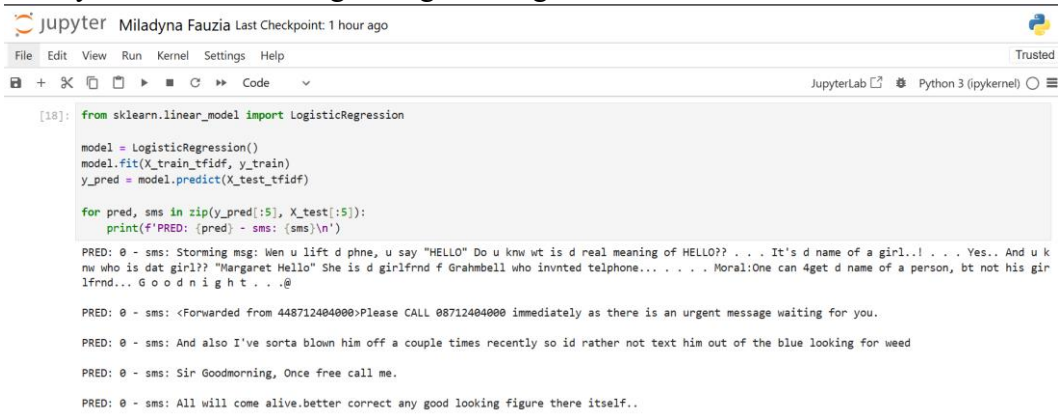
File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[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 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
```

3.5. Binary Classification dengan Logistic Regression



3.6. Evaluation Metrics pada Binary Classification Task

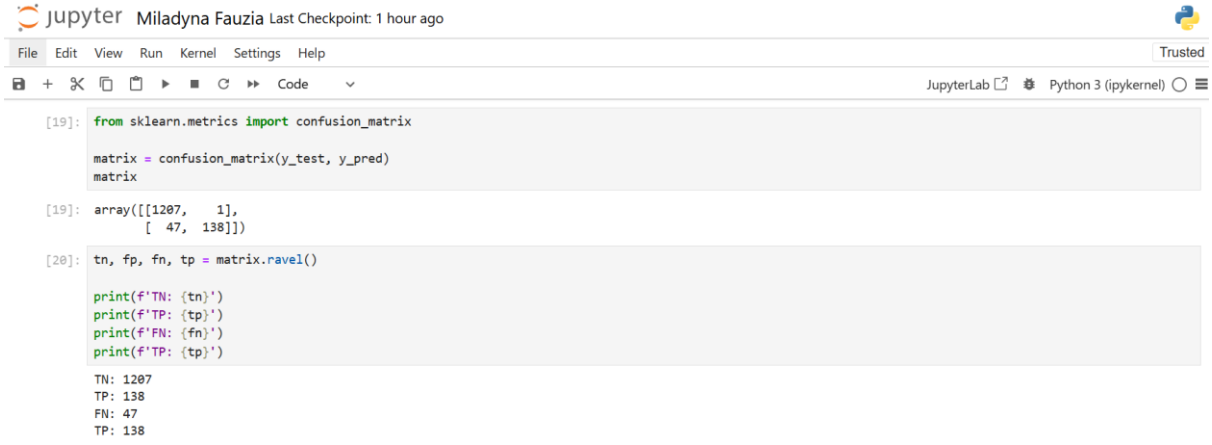
- Confusion Matrix
- Accuracy
- Precision dan Recall
- F1 Score
- ROC

Terminologi Dasar

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

3.7. Pengenalan Confusion Matrix

Confusion Matrix seringkali juga dikenal sebagai error matrix

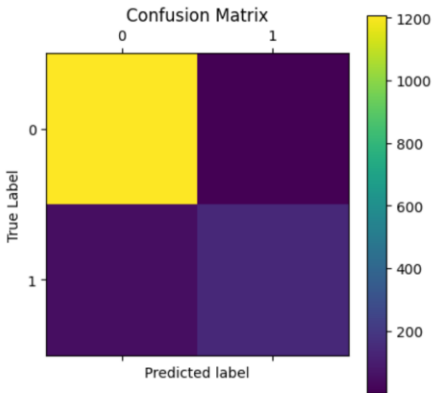


jupyter Miladyna Fauzia Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

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

      plt.matshow(matrix)
      plt.colorbar()

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



3.8. Pengenalan Accuracy Score

Accurary mengukur porsi dari hasil prediksi yang tepat

$$Accurary = \frac{TP + TN}{TP + TN + FP + FN} = \frac{correct}{total}$$

jupyter Miladyna Fauzia Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

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

      accuracy_score(y_test, y_pred)

[22]: 0.9655419956927495
```

3.9. Pengenalan Precision dan Recall

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

Precission or Positive Predictive Value (PPV)

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

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

      precision_score(y_test, y_pred)

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

Recall or True Positive Rate (TPR) or Sensitivity

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

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

      recall_score(y_test, y_pred)

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

3.10. Pengenalan F1 Score | F1 Measure

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

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

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

      f1_score(y_test, y_pred)

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

3.11. Pengenalan ROC | Receiver Operating Characteristic

jupyter Miladyna Fauzia Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[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()
```