APPENDIX

IMPORT LIBRARY

```
1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. from sklearn.metrics import mean_squared_error,
   mean_absolute_percentage_error
```

IMPORT DATA AND VIEW

```
1. df = pd.read_excel('/content/drive/MyDrive/jawa2016.xlsx',
   sheet_name='Clean')    # Cari & copy path file yang sudah
   diupload ke gdrive
2. df = df.drop(['Level', 'Trend', 'Forecast', 'Error'],
    axis=1)
3. print(df.shape)
    df.head()
```

ASSIGN MAIN DATA

```
1. y = df['Magnitude']
```

INPUT AMOUNT OF PREDICTION

```
1. k = int(input('Masukkan jumlah periode yang akan diprediksi
   : '))
```

FUNCTION DES HOLT

```
1. # Make function of Holt exponential smoothing (double
   parameters---> alpha & beta)
2. def double_exponential_smoothing(y, alpha, beta, n_preds=k):
3.     n_record = y.shape[0]
4.     results = np.zeros(n_record + n_preds)
5.
6.     # first value remains the same as series y,
7.     # as there is no history to learn from;
8.     # and the initial trend is the slope/difference
9.     # between the first two value of the series y
10.
11.      levels = [0.0, y[1]]          # input manual (nilai =
   data Magnitude baris ke-2)
12.      trends = [0.0, y[1] - y[0]]    # rumus ---> data
   Magnitude baris kedua dikurangi baris pertama
13.      forecasts = [0.0, 0.0]           # input manual (nilai
   0)
14.
15.      for t in range(2, n_record):
16.          value = y[t]
17.
18.          level = (alpha * value) + ((1 - alpha) * (levels[t-
   1] + trends[t-1]))     # sesuai rumus Level
19.          levels.append(level)
20.
21.          trend = (beta * (levels[t] - levels[t-1])) + ((1 -
   beta) * trends[t-1])   # sesuai rumus Trend
```

```
22.          trends.append(trend)
23.
24.          result = levels[t-1] + trends[t-1]          # sesuai
   rumus Forecast (nilai k = 0)
25.          forecasts.append(result)
26.
27.      # for forecasting beyond the first new point,
28.      # the level and trend is all fixed
29.      if n_preds > 1:
30.          forecasts[n_record+1 :] = levels[-1] +
   (np.arange(1, n_preds + 1) * trends[-1])      # sesuai rumus
   Forecast (nilai k > 0)
31.   return forecasts
```

FIND BEST MAPE

```
1. best_mape = np.inf
2. best_config_mape = []
3.
4. n_preds = k
5.
6. for alpha in np.arange(0.00, 1.02, 0.02):
7.     for beta in np.arange(0.00, 1.02, 0.02):
8.         levels = [0.0, y[1]]
9.         trends = [0.0, y[1] - y[0]]
10.          forecasts = [0.0, 0.0,]
11.
12.          n_record = y.shape[0]
13.          results = np.zeros(n_record + n_preds)
14.
15.          for t in range(2, n_record):
16.              value = y[t]
17.
18.              level = (alpha * value) + ((1 - alpha) *
   (levels[t-1] + trends[t-1]))
19.              levels.append(level)
20.
21.              trend = (beta * (levels[t] - levels[t-1])) +
   ((1 - beta) * trends[t-1])
22.              trends.append(trend)
23.
24.              result = levels[t-1] + trends[t-1]
25.              forecasts.append(result)
26.
27.          if n_preds > 1:
28.              forecasts[n_record + 1:] = levels[-1] +
   (np.arange(1, n_preds + 1) * trends[-1])
29.
30.          mape =
   mean_absolute_percentage_error(y[2:n_record+1],
   forecasts[2:n_record])*100
31.          cfgm = [alpha, beta, mape]
32.
33.          if mape < best_mape:
34.              best_mape = mape
35.              cfg_mape = [alpha, beta, best_mape]
36.              best_config_mape = cfg_mape
```

```
37.  print('Best alpha, beta and MAPE are : ', best_config_mape)
```

PLOT DATA

```
1. def plotting_double_exponential_smoothing(y, alpha, beta) :
2.     plt.figure(figsize=(20, 8))
3.     mape_results = double_exponential_smoothing(y,
   best_config_mape[0], best_config_mape[1])
4.     # rmse_results = double_exponential_smoothing(y,
   best_config_rmse[0], best_config_rmse[1])
5.
6.     plt.plot(y, label='Actual', color='blue', marker='o')
7.     plt.plot(mape_results, label='Forecast MAPE',
   color='red', marker='o', linewidth=3)
8.    # plt.plot(rmse_results, label='Forecast RMSE',
   color='green', marker='x')
9.
10.      plt.legend(loc='best')
11.      plt.axis('tight')
12.      plt.title('Jawa 2016 : Actual -vs- Forecast')
13.   plt.grid(True)
```

PLOTTING

```
1. plotting_double_exponential_smoothing(y, alpha, beta)
```

Similarity Report

PAPER NAME

17.K1.0057.docx

WORD COUNT	CHARACTER COUNT
6636 Words	**29571 Characters**
PAGE COUNT	FILE SIZE
45 Pages	**1.2MB**
SUBMISSION DATE	REPORT DATE
Jul 8, 2022 3:45 PM GMT+7	**Jul 8, 2022 3:46 PM GMT+7**

5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 1% Internet database
- Crossref database
- 3% Submitted Works database
- 3% Publications database
- Crossref Posted Content database

Summary