```
!pip install openpyxl xgboost
```

```
Requirement already satisfied: openpyxl in /usr/local/lib/python3.10/dist-packages (3.1.5)
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.10/dist-packages (from openpyxl) (1.1.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.11.4)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from scipy.stats.mstats import winsorize
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, mean_absolute_percentage_error

```

1. Data Understanding

```
!curl -L -o FINAL_DATASET_with_Humidity_and_Station.xlsx "https://gitlab.com/JPratama7/wa-bot-be/-/raw/main/FINAL_DATASET_with_Humidi
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 35995  100 35995    0     0   114k      0 --:--:-- --:--:-- --:--:--  114k
```

```
# Path ke file Excel
file_path = '/content/FINAL_DATASET_with_Humidity_and_Station.xlsx'

df = pd.read_excel(file_path)
```

```
df
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Hum
0	Cilacap	2013	1322	1255.75075	9.498871	Kacang Tanah	Meteorologi, Cilacap	
1	Banyumas	2013	1671	2172.471503	13.001026	Kacang Tanah	NaN	
2	Purbalingga	2013	731	780.888185	10.682465	Kacang Tanah	NaN	
3	Banjarnegara	2013	2278	1970.754694	8.65125	Kacang Tanah	NaN	
4	Kebumen	2013	2202	1938.738197	8.804442	Kacang Tanah	Sempor, Kebumen	
...	...	...	...	...	...	...	...	
732	Kota Surakarta	2022	27.00	156.00	57.78	Padi	NaN	
733	Kota Salatiga	2022	650.00	3614.00	55.60	Padi	NaN	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 737 entries, 0 to 736
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Kabupaten      737 non-null    object 
 1   Tahun          737 non-null    int64  
 2   Luas Panen     737 non-null    object 
 3   Hasil Panen    737 non-null    object 
 4   Produktivitas  737 non-null    object 
 5   Tanaman        737 non-null    object 
 6   Station        256 non-null    object 
 7   Humidity       256 non-null    float64
dtypes: float64(1), int64(1), object(6)
memory usage: 46.2+ KB
```

```
df = df.query("Tanaman == 'Jagung'")
```

```python
df
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	H
144	Cilacap	2014	2682	15278.107	56.96535	Jagung	Meteorologi, Cilacap	
145	Banyumas	2014	2683	14219.555	52.998714	Jagung	NaN	
146	Purbalingga	2014	5861	31800.95	54.258574	Jagung	NaN	
147	Banjarnegara	2014	14167	78989.632	55.756075	Jagung	NaN	
148	Kebumen	2014	4221	23414.732	55.472002	Jagung	Sempor, Kebumen	
149	Purworejo	2014	2381	14935.1	62.726165	Jagung	NaN	
150	Wonosobo	2014	24461	97420.303	39.826787	Jagung	Wadaslintang, Wonosobo	
151	Magelang	2014	10970	59356.475	54.107999	Jagung	SMPK. Borobudur, Magelang	
152	Boyolali	2014	26933	136434.02	50.656822	Jagung	NaN	
153	Klaten	2014	11178	82934.78	74.19465	Jagung	NaN	
154	Sukoharjo	2014	2210	18497.964	83.701195	Jagung	NaN	
155	Wonogiri	2014	53078	304048.04	57.283251	Jagung	SMPK. Selogiri, Wonogiri	
156	Karanganyar	2014	5001	35295.336	70.576557	Jagung	NaN	
157	Sragen	2014	15323	97011.006	63.310713	Jagung	NaN	
158	Grobogan	2014	105447	590775.63	56.025836	Jagung	NaN	
159	Blora	2014	47199	244814.552	51.868589	Jagung	NaN	
160	Rembang	2014	26948	128384.842	47.641696	Jagung	NaN	
161	Pati	2014	20751	126410.688	60.917878	Jagung	SMPK. Rendole, Pati	
162	Kudus	2014	2792	17063.72	61.116476	Jagung	SMPK. Colo, Kudus	
163	Jepara	2014	6752	52162.414	77.25476	Jagung	NaN	
164	Demak	2014	26082	192155.504	73.673608	Jagung	NaN	
165	Semarang	2014	13589	71486.1	52.605858	Jagung	Klimatologi, Semarang	
166	Semarang	2014	13589	71486.1	52.605858	Jagung	SI Ungaran, Semarang	
167	Temanggung	2014	22865	104520.865	45.716191	Jagung	NaN	

```python
df['Kabupaten'] = df['Kabupaten'].str.replace('Kab. ', '')
df['Kabupaten'] = df['Kabupaten'].str.replace('Kabupaten ', '')
```

```
<ipython-input-9-a905a613c6e2>:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Kabupaten'] = df['Kabupaten'].str.replace('Kab. ', '')
<ipython-input-9-a905a613c6e2>:2: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Kabupaten'] = df['Kabupaten'].str.replace('Kabupaten ', '')
```

```python
df[df['Kabupaten'].str.contains('Kota', case=False, na=False)]
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity
174	Kota Magelang	2014	0	0	0	Jagung	NaN	NaN
175	Kota Surakarta	2014	0	0	0	Jagung	NaN	NaN
176	Kota Salatiga	2014	196	514.146	26.231939	Jagung	NaN	NaN
177	Kota Semarang	2014	626	1566.147	25.018323	Jagung	NaN	NaN

```
df[df['Kabupaten'].str.contains('Kabupaten', case=False, na=False)]
```

Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity

```
df[df['Kabupaten'].str.contains('Kab. ', case=False, na=False)]
```

Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity

```
df.drop(columns=['Station'], inplace=True)
```

```
<ipython-input-13-95b6067be5a9>:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df.drop(columns=['Station'], inplace=True)
```

```
df.isnull().sum()
```

```
Kabupaten        0
Tahun            0
Luas Panen       0
Hasil Panen      0
Produktivitas    0
Tanaman          0
Humidity        25
dtype: int64
```

```
df.isna().sum()
```

```
Kabupaten        0
Tahun            0
Luas Panen       0
Hasil Panen      0
Produktivitas    0
Tanaman          0
Humidity        25
dtype: int64
```

2. Pre-Processing Data

2.1 Mengubah Tipe Data Variabel Menjadi Integer

```
# Membuat function change data type
def clean_data(value):
    if not isinstance(value, str):
      if isinstance(value, int):
        return float(value)
      if isinstance(value, float):
        return value
      return np.nan
    if value == '-':
        return np.nan

    rb = value.split(",", 1)
    if len(rb) > 1:
        value = rb[0] + "." + rb[1]
    elif len(rb) == 1:
        value = rb[0]
    return float(value.replace(',', '.').replace(' ', '').strip())
```

```
df['Luas Panen'] = df['Luas Panen'].apply(clean_data)
df['Hasil Panen'] = df['Hasil Panen'].apply(clean_data)
df['Produktivitas'] = df['Produktivitas'].apply(clean_data)
```

```
<ipython-input-17-1c2a5001d4df>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Luas Panen'] = df['Luas Panen'].apply(clean_data)
<ipython-input-17-1c2a5001d4df>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Hasil Panen'] = df['Hasil Panen'].apply(clean_data)
<ipython-input-17-1c2a5001d4df>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Produktivitas'] = df['Produktivitas'].apply(clean_data)
```

```
df.head()
```

```
       Kabupaten  Tahun  Luas Panen  Hasil Panen  Produktivitas  Tanaman  Humidity
144      Cilacap   2014      2682.0    15278.107      56.965350   Jagung      82.0
145     Banyumas   2014      2683.0    14219.555      52.998714   Jagung       NaN
146  Purbalingga   2014      5861.0    31800.950      54.258574   Jagung       NaN
147 Banjarnegara   2014     14167.0    78989.632      55.756075   Jagung       NaN
148      Kebumen   2014      4221.0    23414.732      55.472002   Jagung      85.0
```

2.2 Interpolate Data NaN

```
df.isna().sum()
```

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         25
dtype: int64
```

```
df['Luas Panen'] = df['Luas Panen'].interpolate(method='spline', order=2)
df['Hasil Panen'] = df['Hasil Panen'].interpolate(method='spline', order=2)
df['Produktivitas'] = df['Produktivitas'].interpolate(method='spline', order=2)
df['Humidity'] = df['Humidity'].interpolate(method='spline', order=2)
```

```
<ipython-input-20-ae5dbdddba06>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Luas Panen'] = df['Luas Panen'].interpolate(method='spline', order=2)
<ipython-input-20-ae5dbdddba06>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Hasil Panen'] = df['Hasil Panen'].interpolate(method='spline', order=2)
<ipython-input-20-ae5dbdddba06>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Produktivitas'] = df['Produktivitas'].interpolate(method='spline', order=2)
<ipython-input-20-ae5dbdddba06>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Humidity'] = df['Humidity'].interpolate(method='spline', order=2)
```

```
df.isna().sum()
```

```
Kabupaten        0
Tahun            0
Luas Panen       0
Hasil Panen      0
Produktivitas    0
Tanaman          0
Humidity         0
dtype: int64
```

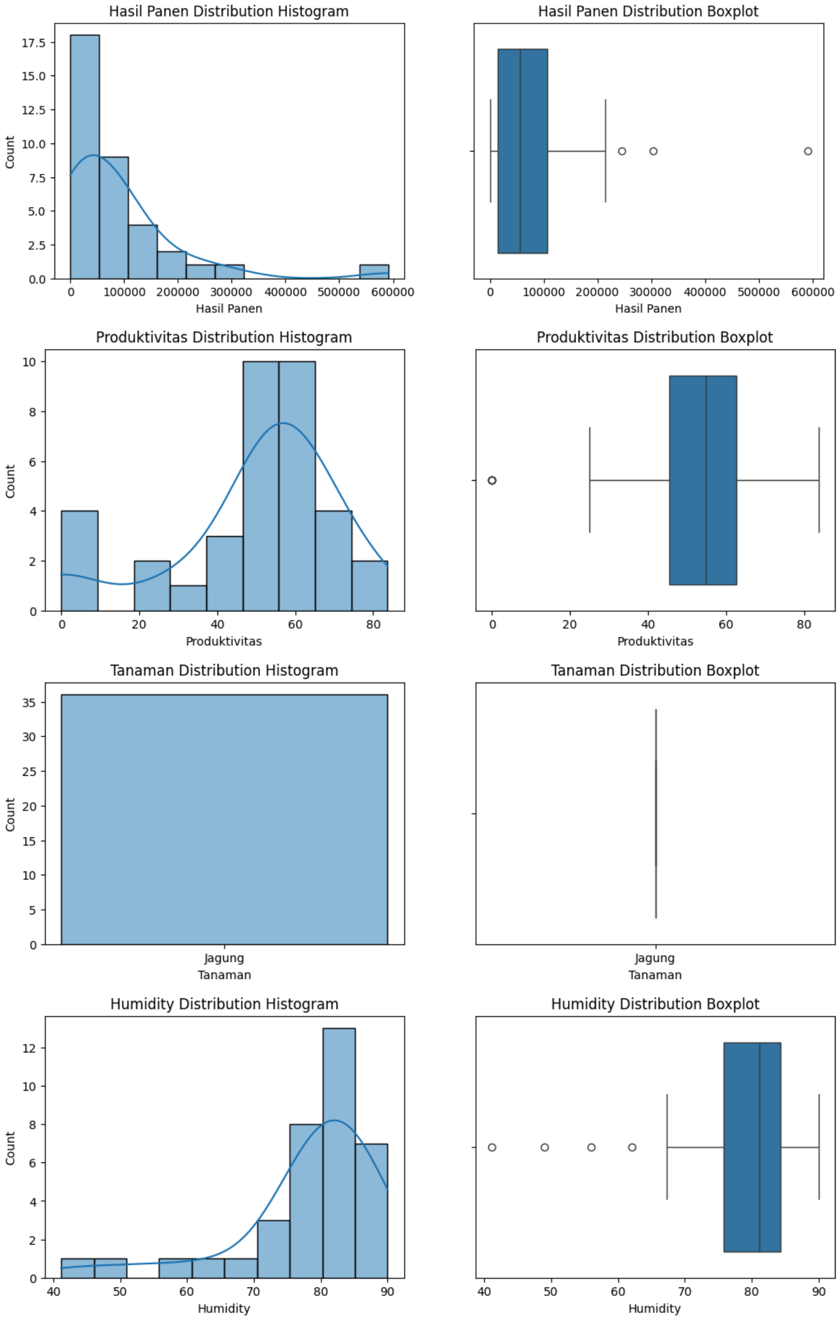
2.3 Handling outlier

```
# Fungsi untuk menampilkan boxplot dan histogram
def num_dist(data, var):
    fig, ax = plt.subplots(1, 2, figsize=(12, 4))

    sns.histplot(data=data, x=var, kde=True, ax=ax[0])
    sns.boxplot(data=data, x=var, ax=ax[1])
    ax[0].set_title(f"{var} Distribution Histogram")
    ax[1].set_title(f"{var} Distribution Boxplot")

    plt.show()

df_var = df.columns
for var in df_var:
    num_dist(df, var)
```



Hasil Panen Distribution Histogram

Hasil Panen Distribution Boxplot

Produktivitas Distribution Histogram

Produktivitas Distribution Boxplot

Tanaman Distribution Histogram

Tanaman Distribution Boxplot

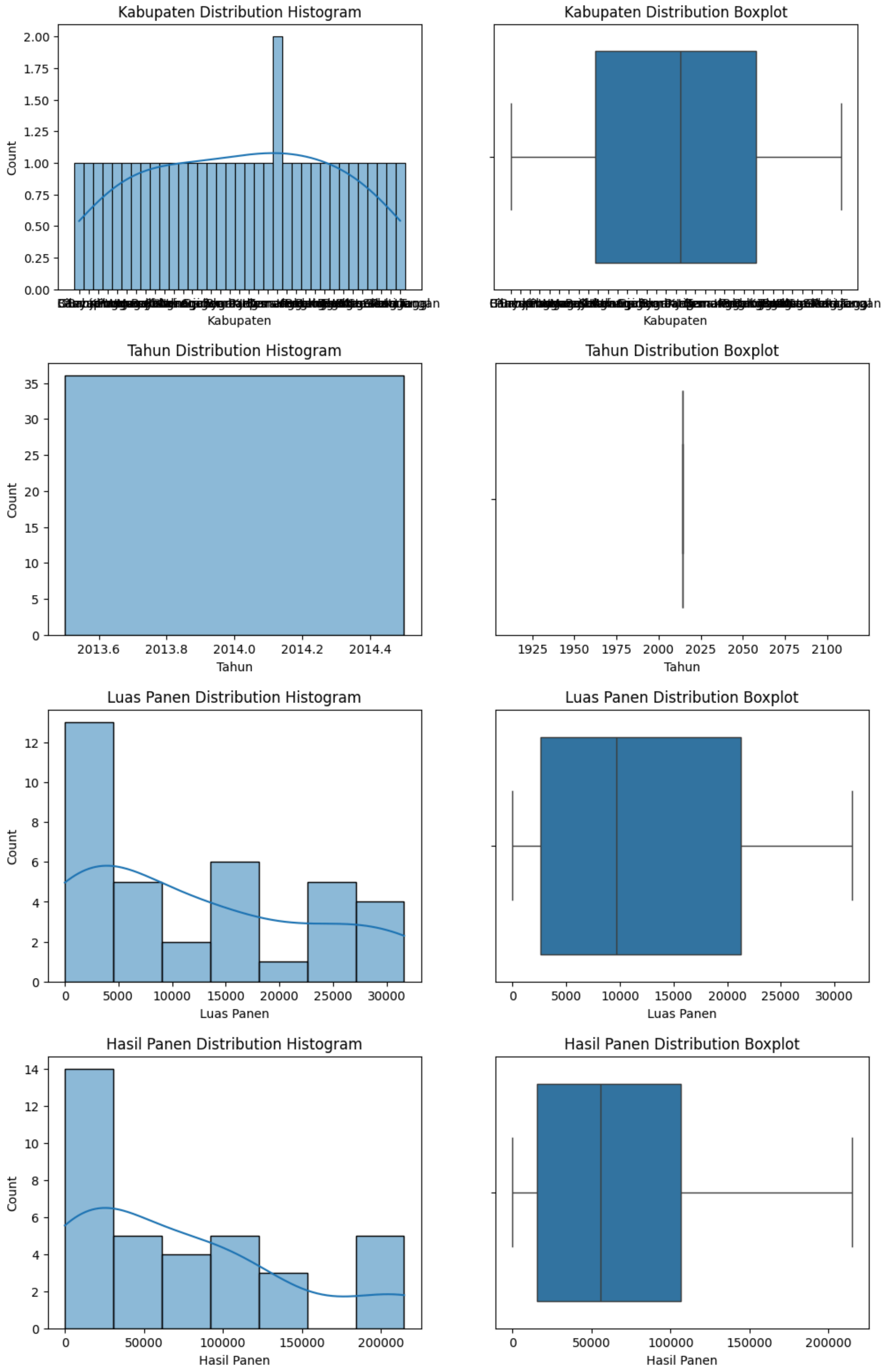
Humidity Distribution Histogram

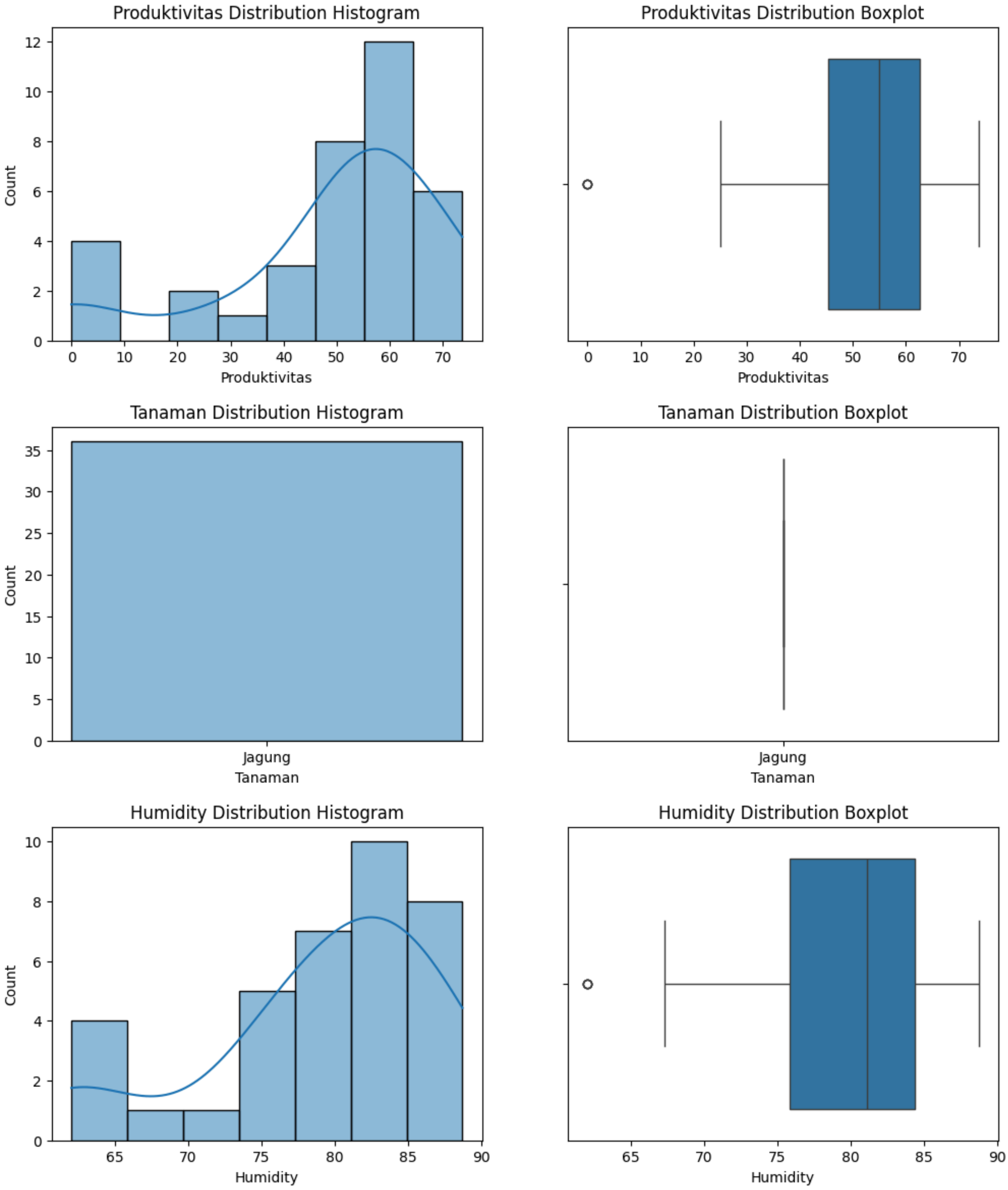
Humidity Distribution Boxplot

```
# Variabel yang terdapat outlier
outlier_var = ['Humidity', 'Produktivitas', 'Hasil Panen', 'Luas Panen']
threshold = 0.1

# Menggunakan treatment Winsorize untuk menghapus Outlier
for var in outlier_var:
    df.loc[:, var] = winsorize(df[var], limits=[threshold, threshold])
```

```
df_var = df.columns
for var in df_var:
    num_dist(df, var)
```


Kabupaten Distribution Histogram
Count
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Kabupaten
Kabupaten Distribution Boxplot
Kabupaten
Tahun Distribution Histogram
Count
35
30
25
20
15
10
5
0
2013.6
2013.8
2014.0
2014.2
2014.4
Tahun
Tahun Distribution Boxplot
1925
1950
1975
2000
2025
2050
2075
2100
Tahun
Luas Panen Distribution Histogram
Count
12
10
8
6
4
2
0
0
5000
10000
15000
20000
25000
30000
Luas Panen
Luas Panen Distribution Boxplot
0
5000
10000
15000
20000
25000
30000
Luas Panen
Hasil Panen Distribution Histogram
Count
14
12
10
8
6
4
2
0
0
50000
100000
150000
200000
Hasil Panen
Hasil Panen Distribution Boxplot
0
50000
100000
150000
200000
Hasil Panen



```
numerical_columns = ['Luas Panen', 'Hasil Panen', 'Produktivitas', 'Humidity']


description = df[numerical_columns].describe().T

# Menambahkan percentiles ke deskripsi
description['25%'] = df[numerical_columns].quantile(0.25)
description['50%'] = df[numerical_columns].quantile(0.50)
description['75%'] = df[numerical_columns].quantile(0.75)

# Menyusun ulang kolom agar sesuai dengan format tabel di gambar
description = description[['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max']]
description
```

	count	mean	std	min	25%	50%	75%	max
Luas Panen	36.0	12244.111111	10949.041205	0.000000	2606.750000	9682.500000	21279.500000	31607.000000
Hasil Panen	36.0	72979.843389	69643.319248	0.000000	15192.355250	55759.444500	106230.696250	214636.920000
Produktivitas	36.0	49.432562	21.278417	0.000000	45.338016	54.865288	62.594229	73.673608
Humidity	36.0	79.177376	7.897786	62.056158	75.847062	81.091185	84.369629	88.728318

Next steps: Generate code with description View recommended plots

2.4 Encoding

```
encoder = LabelEncoder()
df['Tanaman'] = encoder.fit_transform(df['Tanaman'])
df['Kabupaten'] = encoder.fit_transform(df['Kabupaten'])
```

```
<ipython-input-25-d457133ed473>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Tanaman'] = encoder.fit_transform(df['Tanaman'])
<ipython-input-25-d457133ed473>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Kabupaten'] = encoder.fit_transform(df['Kabupaten'])
```

2.5 Visualisasi Data

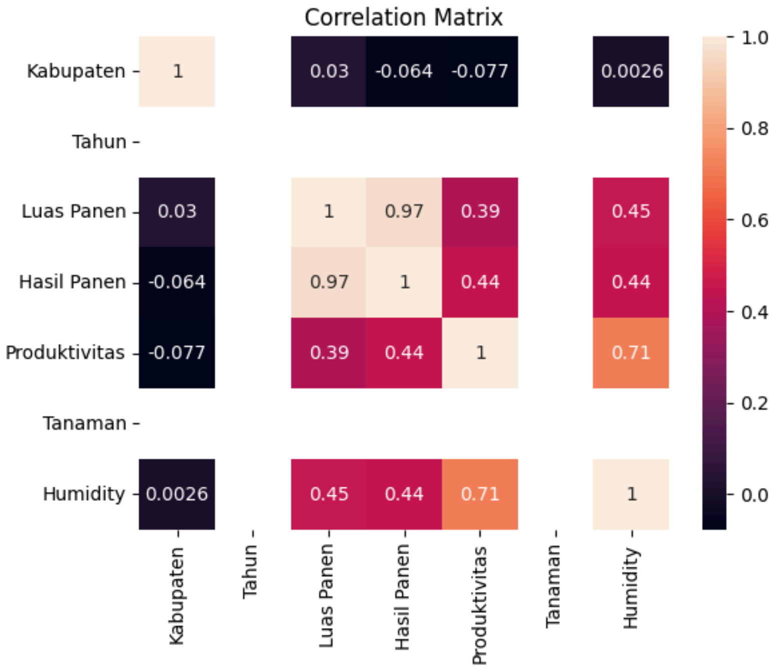
2.5 Correlation

```
df.corr()
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Humidity
Kabupaten	1.000000	NaN	0.029784	-0.063614	-0.076609	NaN	0.002649
Tahun	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Luas Panen	0.029784	NaN	1.000000	0.969558	0.394023	NaN	0.451310
Hasil Panen	-0.063614	NaN	0.969558	1.000000	0.444002	NaN	0.441243
Produktivitas	-0.076609	NaN	0.394023	0.444002	1.000000	NaN	0.711231
Tanaman	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Humidity	0.002649	NaN	0.451310	0.441243	0.711231	NaN	1.000000

```
sns.heatmap(df.corr(), annot =True)
plt.title('Correlation Matrix')
```

```
Text(0.5, 1.0, 'Correlation Matrix')
```



3. Modelling

```
from sklearn.model_selection import train_test_split

x = df.drop(["Hasil Panen", "Kabupaten", "Tahun", "Tanaman"], axis=1)
y = df["Hasil Panen"]

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=5)

print("x_train :",x_train.shape)
print("x_test :",x_test.shape)
print("y_train :",y_train.shape)
print("y_test :",y_test.shape)
```

```
x_train : (28, 3)
x_test : (8, 3)
y_train : (28,)
y_test : (8,)
```

3.1 Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators = 11)
rf_model.fit(x_train,y_train)
rf_predict = rf_model.predict(x_test)
```

```
rf_model.score(x_test,y_test)
```

```
0.9268821622549585
```

```
feature_importance = pd.Series(rf_model.feature_importances_, index=x.columns)
feature_importance.sort_values(ascending=False)
```

```
Luas Panen       0.962704
Humidity         0.019704
Produktivitas    0.017591
dtype: float64
```

```python
# Get feature importances
feature_importances = rf_model.feature_importances_
features = x.columns

# Create a DataFrame for better visualization
feature_importance_df = pd.DataFrame({
    'Feature': features,
    'Importance': feature_importances
})

# Sort the DataFrame by importance
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Display the feature importances DataFrame
feature_importance_df
```

	Feature	Importance
0	Luas Panen	0.962704
2	Humidity	0.019704
1	Produktivitas	0.017591

```python
# K Fold RF
kf_rf = KFold(n_splits=5, shuffle=True, random_state=42)

rf_r2_scores = []

rf_model_kf = RandomForestRegressor(n_estimators = 11)

for train_index, test_index in kf_rf.split(x):
    x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]

    # Melatih model
    rf_model_kf.fit(x_train_fold, y_train_fold)

    # Memprediksi hasil pada data uji
    y_pred_fold = rf_model_kf.predict(x_test_fold)

    # Menghitung skor R-squared
    r2 = r2_score(y_test_fold, y_pred_fold)
    rf_r2_scores.append(r2)

rf_r2_scores.append(np.mean(rf_r2_scores))
rf_r2_scores.append(np.std(rf_r2_scores))
# Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
print("Skor untuk setiap fold: ", rf_r2_scores)
print("Rata-rata skor R-squared: ", np.mean(rf_r2_scores))
print("Standar deviasi skor R-squared: ", np.std(rf_r2_scores))
```

```
Skor untuk setiap fold:  [0.9193329393075658, 0.8866415570352332, 0.8333368838610791, 0.9292545012740299, 0.8866870449648345, 0.891
Rata-rata skor R-squared:  0.7681332036196651
Standar deviasi skor R-squared:  0.30241727468097135
```

```python
# Evaluation
y_pred = rf_model.predict(x_test)

print("R-Squared : ", r2_score(y_test, y_pred))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, y_pred)))
print("MAE : ", mean_absolute_error(y_test, y_pred))
```

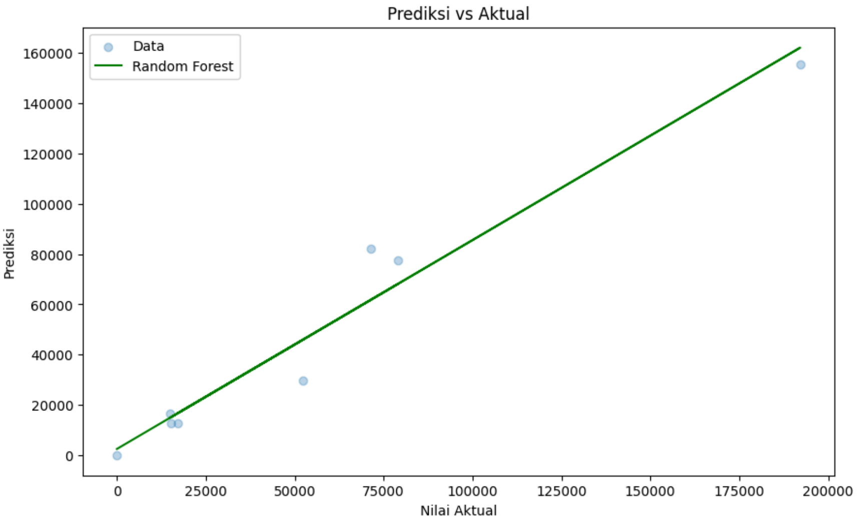
```
R-Squared :  0.9268821622549585
RMSE :  15799.97412700798
MAE :  9963.896488636365
```

```
# Membuat plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3, label='Data')

# Menghitung garis regresi linear
coef = np.polyfit(y_test, y_pred, 1)
poly1d_fn = np.poly1d(coef)

# Menambahkan garis regresi linear pada plot
plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')

plt.xlabel('Nilai Aktual')
plt.ylabel('Prediksi')
plt.title('Prediksi vs Aktual')
plt.legend()
plt.show()
```



3.2 Linear Regression

```
from sklearn.linear_model import LinearRegression

# Inisialisasi model Linear Regression
lr_model = LinearRegression()

# Melatih model
lr_model.fit(x_train, y_train)

# Memprediksi hasil pada data uji
lr_predict = lr_model.predict(x_test)
```

```
# Evaluation
print("R-Squared : ", r2_score(y_test, lr_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, lr_predict)))
print("MAE : ", mean_absolute_error(y_test, lr_predict))

# Mengambil koefisien sebagai feature importance
lr_importance = lr_model.coef_

# Membuat dataframe untuk feature importance
lr_importance_df = pd.DataFrame({
    'Feature': x_train.columns,
    'Importance': lr_importance
})

lr_importance_df
```

```
R-Squared :  0.9602897067915842
RMSE :  11643.84063376389
MAE :  8258.897608119212
```

	Feature	Importance
0	Luas Panen	5.966266
1	Produktivitas	348.663158
2	Humidity	-376.330746

```
# K Fold LR
kf_lr = KFold(n_splits=5, shuffle=True, random_state=42)

lr_r2_scores = []

lr_model_kf = LinearRegression()

for train_index, test_index in kf_lr.split(x):
    x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]

    # Melatih model
    lr_model_kf.fit(x_train_fold, y_train_fold)

    # Memprediksi hasil pada data uji
    y_pred_fold = lr_model_kf.predict(x_test_fold)

    # Menghitung skor R-squared
    r2 = r2_score(y_test_fold, y_pred_fold)
    lr_r2_scores.append(r2)

lr_r2_scores.append(np.mean(lr_r2_scores))
lr_r2_scores.append(np.std(lr_r2_scores))
# Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
print("Skor untuk setiap fold: ", lr_r2_scores)
print("Rata-rata skor R-squared: ", np.mean(lr_r2_scores))
print("Standar deviasi skor R-squared: ", np.std(lr_r2_scores))
```

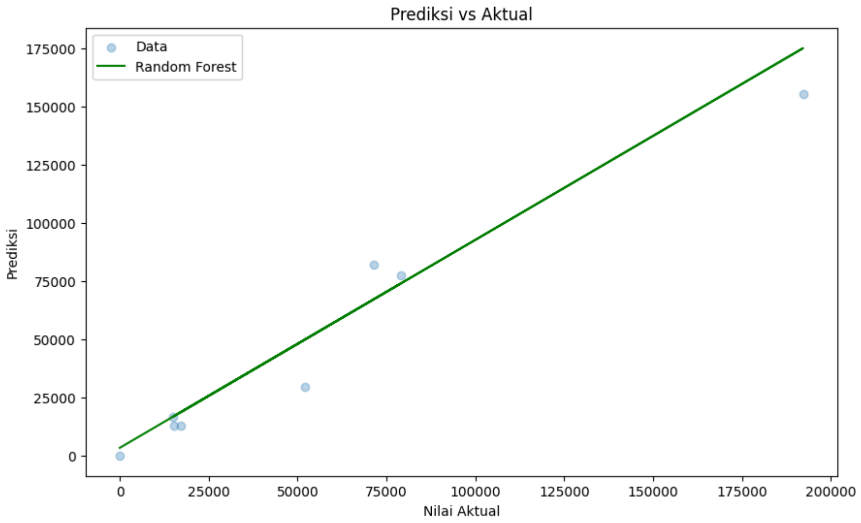
```
Skor untuk setiap fold:  [0.7994373335535405, 0.9515802401892022, 0.9234743980692791, 0.8209172457427636, 0.8858389226169953, 0.876
Rata-rata skor R-squared:  0.7586660050300423
Standar deviasi skor R-squared:  0.29219533301910044
```

```
# Membuat plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3, label='Data')

# Menghitung garis regresi linear
coef = np.polyfit(y_test, lr_predict, 1)
poly1d_fn = np.poly1d(coef)

# Menambahkan garis regresi linear pada plot
plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')

plt.xlabel('Nilai Aktual')
plt.ylabel('Prediksi')
plt.title('Prediksi vs Aktual')
plt.legend()
plt.show()
```



3.3 Support Vector Machine

```
from sklearn.svm import SVR

# Inisialisasi model SVR
svr_model = SVR(C= 1000, degree=3, gamma='auto', kernel='linear')  # Anda bisa menggunakan kernel lain seperti 'linear', 'poly', dll.

# Melatih model
svr_model.fit(x_train, y_train)  # Menggunakan y_train.ravel() untuk mengubah bentuk jika diperlukan

# Memprediksi hasil pada data uji
svr_predict = svr_model.predict(x_test)
```

```
# K Fold SVR
kf_svr = KFold(n_splits=5, shuffle=True, random_state=42)

svr_r2_scores = []

svr_model_kf = SVR(C= 1000, degree=3, gamma='auto', kernel='linear')  # Anda bisa menggunakan kernel lain seperti 'linear', 'poly', dl

for train_index, test_index in kf_svr.split(x):
    x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]

    # Melatih model
    svr_model_kf.fit(x_train_fold, y_train_fold)

    # Memprediksi hasil pada data uji
    y_pred_fold = svr_model_kf.predict(x_test_fold)

    # Menghitung skor R-squared
    r2 = r2_score(y_test_fold, y_pred_fold)
    svr_r2_scores.append(r2)

svr_r2_scores.append(np.mean(svr_r2_scores))
svr_r2_scores.append(np.std(svr_r2_scores))
# Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
print("Skor untuk setiap fold: ", svr_r2_scores)
print("Rata-rata skor R-squared: ", np.mean(svr_r2_scores))
print("Standar deviasi skor R-squared: ", np.std(svr_r2_scores))
```

```
Skor untuk setiap fold:  [0.9177082131957582, 0.9634884428180976, 0.9750514597559203, 0.5452320420100865, 0.7527099817837616, 0.830
Rata-rata skor R-squared:  0.7334652362238226
Standar deviasi skor R-squared:  0.27563852880674
```

```
# Evaluation
print("R-Squared : ", r2_score(y_test, svr_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, svr_predict)))
print("MAE : ", mean_absolute_error(y_test, svr_predict))
```

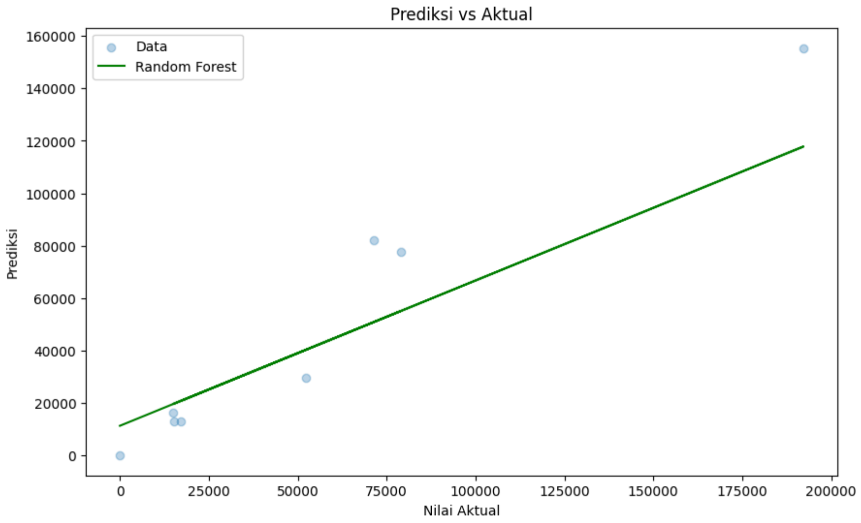
```
R-Squared :  0.7370902130738363
RMSE :  29960.432777712947
MAE :  16997.75725032366
```

```
# Membuat plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3, label='Data')

# Menghitung garis regresi linear
coef = np.polyfit(y_test, svr_predict, 1)
poly1d_fn = np.poly1d(coef)

# Menambahkan garis regresi linear pada plot
plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')

plt.xlabel('Nilai Aktual')
plt.ylabel('Prediksi')
plt.title('Prediksi vs Aktual')
plt.legend()
plt.show()
```



3.4 Lasso Regression

```
from sklearn.linear_model import Lasso

# Inisialisasi model Lasso Regression
lasso_model = Lasso(alpha=0.0001)  # Anda bisa mengatur nilai alpha sesuai kebutuhan

# Melatih model
lasso_model.fit(x_train, y_train)

# Memprediksi hasil pada data uji
lasso_predict = lasso_model.predict(x_test)
```

```
# Evaluation

print("R-Squared : ", r2_score(y_test, lasso_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, lasso_predict)))
print("MAE : ", mean_absolute_error(y_test, lasso_predict))
```

```
R-Squared :  0.9602897066736517
RMSE :  11643.840651053964
MAE :  8258.897617925028
```

```
# K Fold Lasso
kf_lasso = KFold(n_splits=5, shuffle=True, random_state=42)

lasso_r2_scores = []

lasso_model_kf = Lasso(alpha=0.0001)

for train_index, test_index in kf_lasso.split(x):
    x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]

    # Melatih model
    lasso_model_kf.fit(x_train_fold, y_train_fold)

    # Memprediksi hasil pada data uji
    y_pred_fold = lasso_model_kf.predict(x_test_fold)

    # Menghitung skor R-squared
    r2 = r2_score(y_test_fold, y_pred_fold)
    lasso_r2_scores.append(r2)

lasso_r2_scores.append(np.mean(lasso_r2_scores))
lasso_r2_scores.append(np.std(lasso_r2_scores))
# Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
print("Skor untuk setiap fold: ", lasso_r2_scores)
print("Rata-rata skor R-squared: ", np.mean(lasso_r2_scores))
print("Standar deviasi skor R-squared: ", np.std(lasso_r2_scores))
```

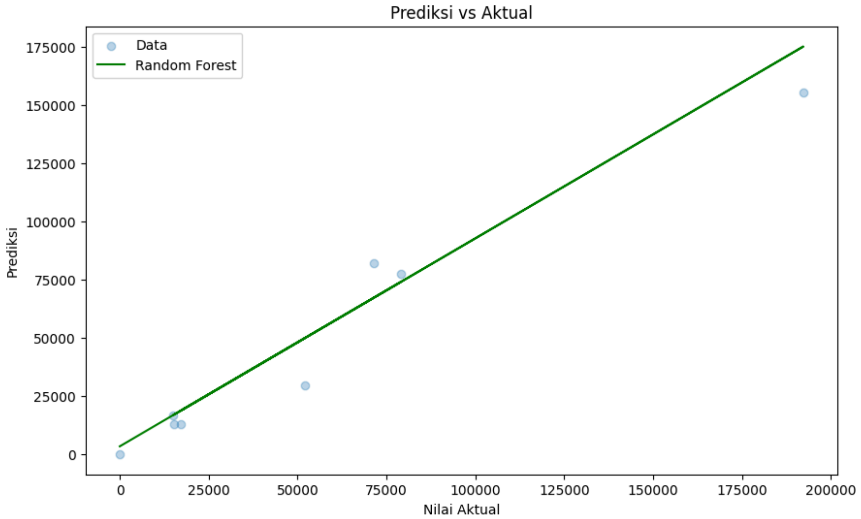
```
Skor untuk setiap fold:  [0.7994373338758171, 0.9515802401845483, 0.9234743981293868, 0.8209172458392526, 0.8858389224808354, 0.8762
Rata-rata skor R-squared:  0.7586660050750478
Standar deviasi skor R-squared:  0.2921953330595492
```

```
# Membuat plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3, label='Data')

# Menghitung garis regresi linear
coef = np.polyfit(y_test, lasso_predict, 1)
poly1d_fn = np.poly1d(coef)

# Menambahkan garis regresi linear pada plot
plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')

plt.xlabel('Nilai Aktual')
plt.ylabel('Prediksi')
plt.title('Prediksi vs Aktual')
plt.legend()
plt.show()
```



3.5 XGBoost Regression

```
import xgboost as xgb

# Inisialisasi model XGBoost
xgb_model = xgb.XGBRegressor(
    n_estimators=1000,
    learning_rate=0.01,
    max_depth=30,
    random_state=42
)

# Melatih model
xgb_model.fit(x_train, y_train)

# Memprediksi hasil pada data uji
xgb_predict = xgb_model.predict(x_test)
```

```
# Evaluation
print("R-Squared : ", r2_score(y_test, xgb_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, xgb_predict)))
print("MAE : ", mean_absolute_error(y_test, xgb_predict))
```

```
R-Squared :  0.8261462563492007
RMSE :  24363.334575674686
MAE :  11990.328346878048
```

```
# K Fold xgb
kf_xgb = KFold(n_splits=5, shuffle=True, random_state=42)

xgb_r2_scores = []

xgb_model_kf = xgb.XGBRegressor(
    n_estimators=1000,
    learning_rate=0.01,
    max_depth=30,
    random_state=42
)

for train_index, test_index in kf_xgb.split(x):
    x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]

    # Melatih model
    xgb_model_kf.fit(x_train_fold, y_train_fold)

    # Memprediksi hasil pada data uji
    y_pred_fold = xgb_model_kf.predict(x_test_fold)

    # Menghitung skor R-squared
    r2 = r2_score(y_test_fold, y_pred_fold)
    xgb_r2_scores.append(r2)

xgb_r2_scores.append(np.mean(xgb_r2_scores))
xgb_r2_scores.append(np.std(xgb_r2_scores))
# Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
print("Skor untuk setiap fold: ", xgb_r2_scores)
print("Rata-rata skor R-squared: ", np.mean(xgb_r2_scores))
print("Standar deviasi skor R-squared: ", np.std(xgb_r2_scores))
```

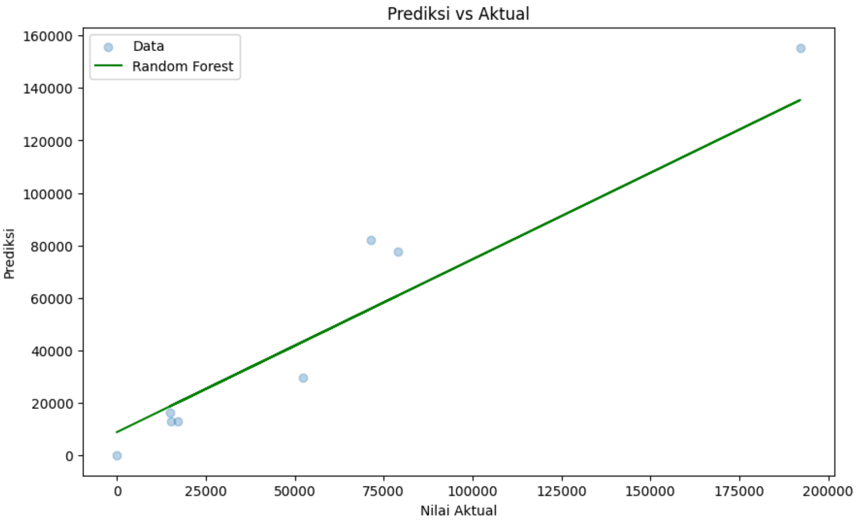
```
Skor untuk setiap fold:  [0.9808756934373716, 0.9238018057401076, 0.9436129062651706, 0.9217371416972686, 0.8445011904225089, 0.922
Rata-rata skor R-squared:  0.7968762977349187
Standar deviasi skor R-squared:  0.31099896119680404
```

```
# Membuat plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3, label='Data')

# Menghitung garis regresi linear
coef = np.polyfit(y_test, xgb_predict, 1)
poly1d_fn = np.poly1d(coef)

# Menambahkan garis regresi linear pada plot
plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')

plt.xlabel('Nilai Aktual')
plt.ylabel('Prediksi')
plt.title('Prediksi vs Aktual')
plt.legend()
plt.show()
```



Evaluasi