```
!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.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_Humid
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 35995  100 35995    0     0  96627      0 --:--:-- --:--:-- --:--:-- 96760
```

```
# 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	

Next steps: Generate code with df View recommended plots

```
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 == 'Ubi Jalar'")
```

```
df
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity
399	Cilacap	2015	228	4942.14	216.760526	Ubi Jalar	Meteorologi, Cilacap	83.(
400	Banyumas	2015	160	1620	101.25	Ubi Jalar	NaN	NaN
401	Purbalingga	2015	98	3603.56	367.710204	Ubi Jalar	NaN	NaN
402	Banjarnegara	2015	140	1681.848	120.132	Ubi Jalar	NaN	NaN
403	Kebumen	2015	58	981.52	169.227586	Ubi Jalar	Sempor, Kebumen	77.(
...	...	...	...	...	...	...	...	..
661	Semarang	2019	10,00	253,00	258,36	Ubi Jalar	SMPK. Balit Getas, Semarang	81.(

Next steps: Generate code with df View recommended plots

```
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 ', '')
```

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

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity
467	Kota Magelang	2018	0	0	0	Ubi Jalar	NaN	NaN
468	Kota Surakarta	2018	0	0	0	Ubi Jalar	NaN	NaN
469	Kota Salatiga	2018	4	103	258.7	Ubi Jalar	NaN	NaN
	Kota					Ubi		

```
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         67
dtype: int64
```

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

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         67
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
399	Cilacap	2015	228.0	4942.140	216.760526	Ubi Jalar	83.0
400	Banyumas	2015	160.0	1620.000	101.250000	Ubi Jalar	NaN
401	Purbalingga	2015	98.0	3603.560	367.710204	Ubi Jalar	NaN

Next steps: Generate code with df   View recommended plots

2.2 Interpolate Data NaN

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

```
Kabupaten         0
Tahun             0
Luas Panen       12
Hasil Panen      12
Produktivitas    12
Tanaman           0
Humidity         67
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)
```

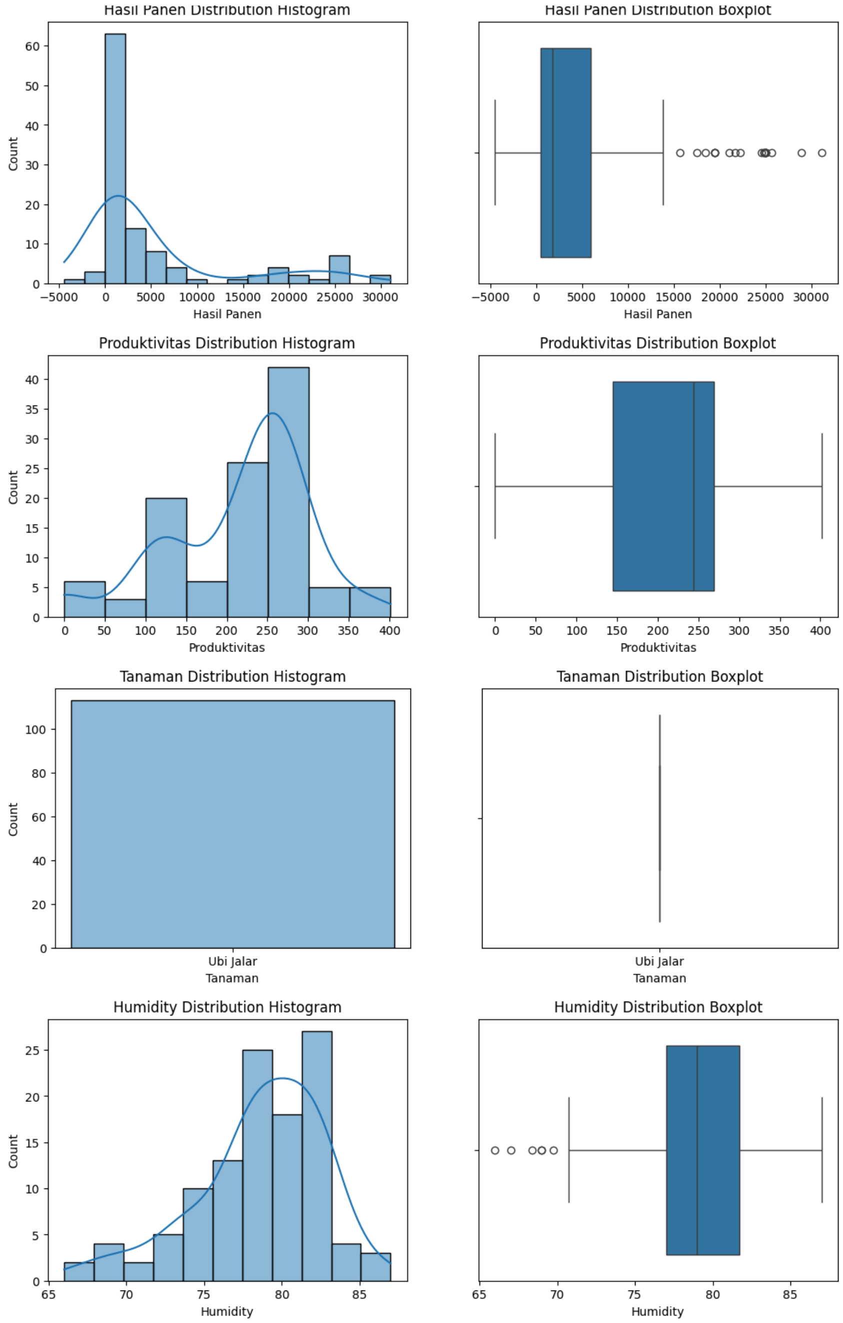
```
<ipython-input-20-ae5dbdddba06>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

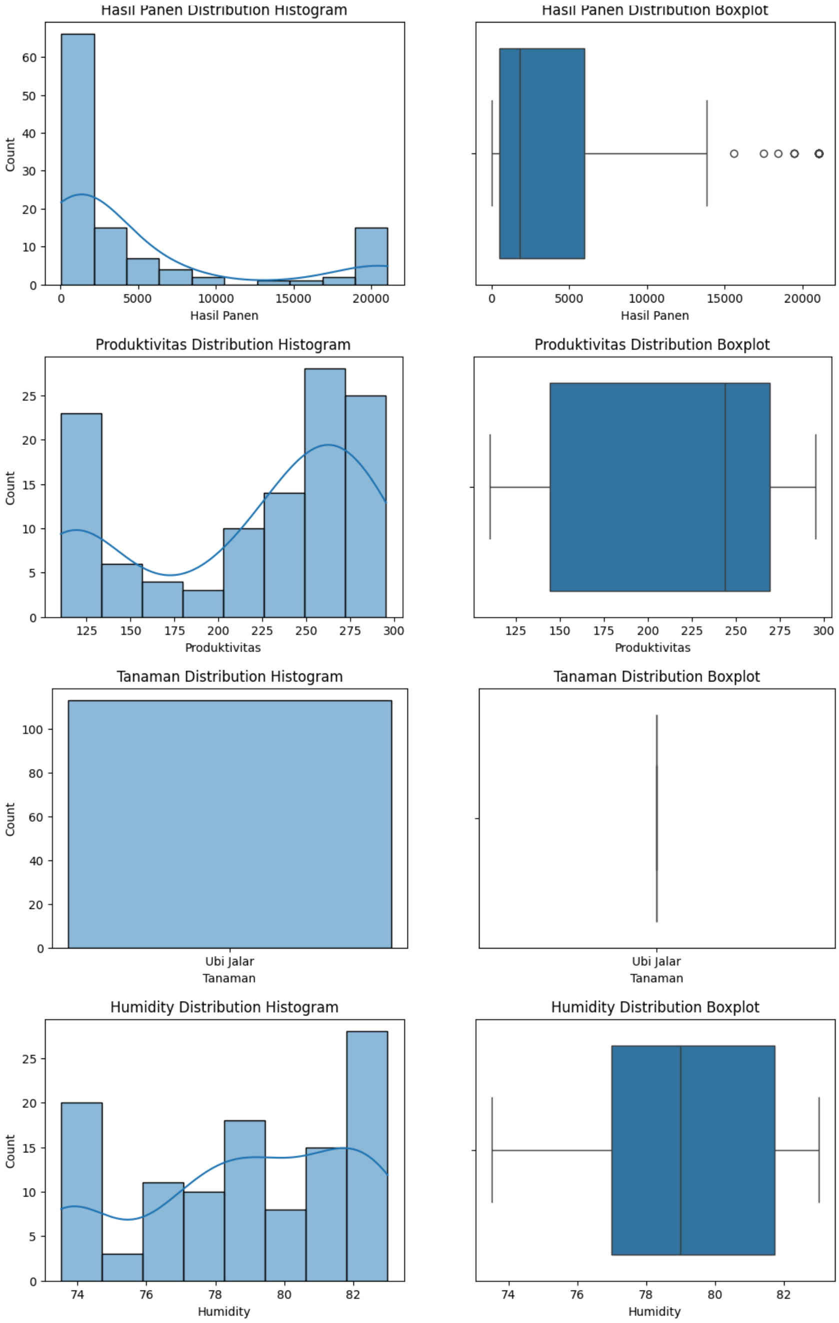
Tanaman Distribution Boxplot

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)
```


Hasil Panen Distribution Histogram
Count
0
10
20
30
40
50
60
0
5000
10000
15000
20000
Hasil Panen
Hasil Panen Distribution Boxplot
0
5000
10000
15000
20000
Hasil Panen
Produktivitas Distribution Histogram
Count
0
5
10
15
20
25
125
150
175
200
225
250
275
300
Produktivitas
Produktivitas Distribution Boxplot
125
150
175
200
225
250
275
300
Produktivitas
Tanaman Distribution Histogram
Count
0
20
40
60
80
100
Ubi Jalar
Tanaman
Tanaman Distribution Boxplot
Ubi Jalar
Tanaman
Humidity Distribution Histogram
Count
0
5
10
15
20
25
74
76
78
80
82
Humidity
Humidity Distribution Boxplot
74
76
78
80
82
Humidity

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

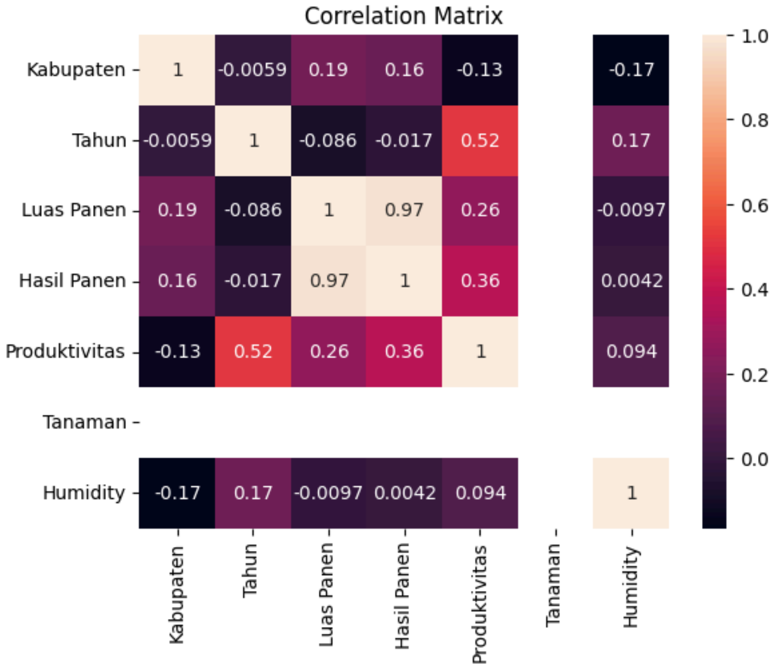
2.1 Correlation

```
df.corr()
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Humidi
Kabupaten	1.000000	-0.005851	0.189406	0.161429	-0.130950	NaN	-0.1654
Tahun	-0.005851	1.000000	-0.086130	-0.016892	0.516933	NaN	0.1676
Luas Panen	0.189406	-0.086130	1.000000	0.973502	0.263130	NaN	-0.0097
Hasil Panen	0.161429	-0.016892	0.973502	1.000000	0.363656	NaN	0.0042
Produktivitas	-0.130950	0.516933	0.263130	0.363656	1.000000	NaN	0.0940
Tanaman	NaN	NaN	NaN	NaN	NaN	NaN	N

```
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"]

# Splitting data set - 25% test dataset and 75%

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 : (90, 3)
x_test : (23, 3)
y_train : (90,)
```

```
y_test : (23,)
```

3.1 Random Forest Regression

```python
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)
```

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

```
0.8520592623836004
```

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

```
Luas Panen       0.992388
Produktivitas    0.007299
Humidity         0.000313
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	0.992388
1	Produktivitas	0.007299
2	Humidity	0.000313

Next steps: Generate code with `feature_importance_df` View recommended plots

```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))
```

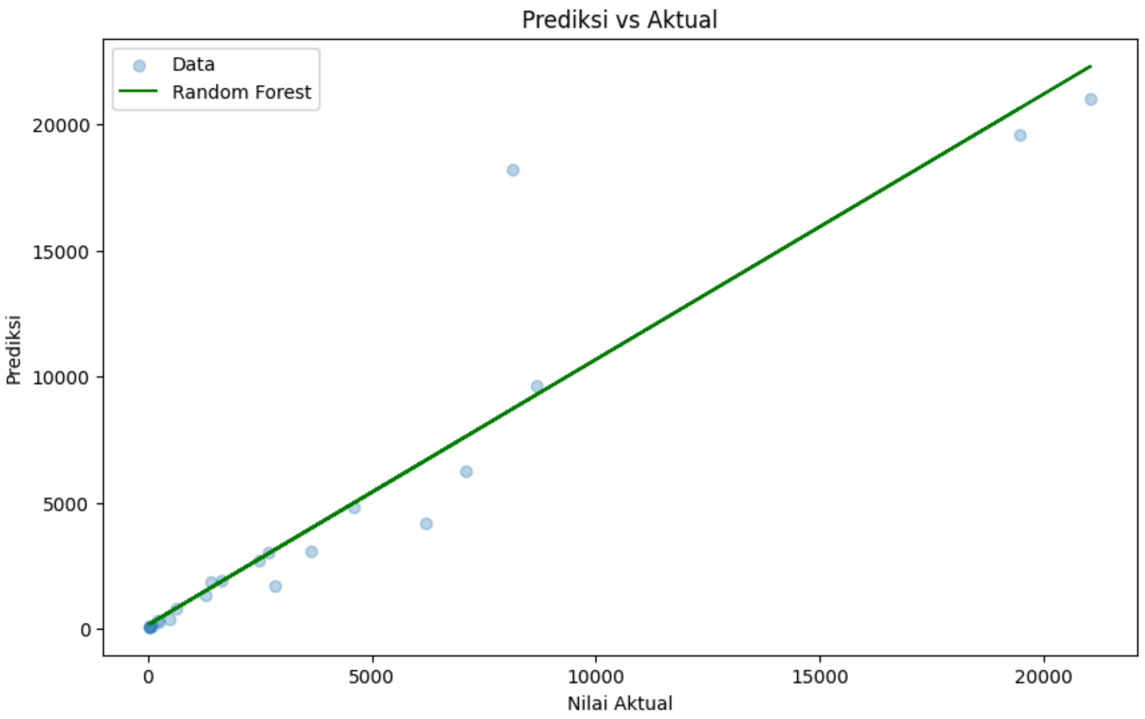
```
Skor untuk setiap fold:  [0.9941398455208373, 0.9924428288870127, 0.8823924865912809, 0.9964749834928758, 0.9520282496139985, 0.963
Rata-rata skor R-squared:  0.8315599416382329
```

```
Standar deviasi skor R-squared:  0.32528437976667557
```

```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 :  0.8520592623836004
MAE :  773.7191853163163
```

```python
# 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

```python
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 :  0.8558295435128753
RMSE :  2151.668453161631
MAE :  983.8210232790678
```

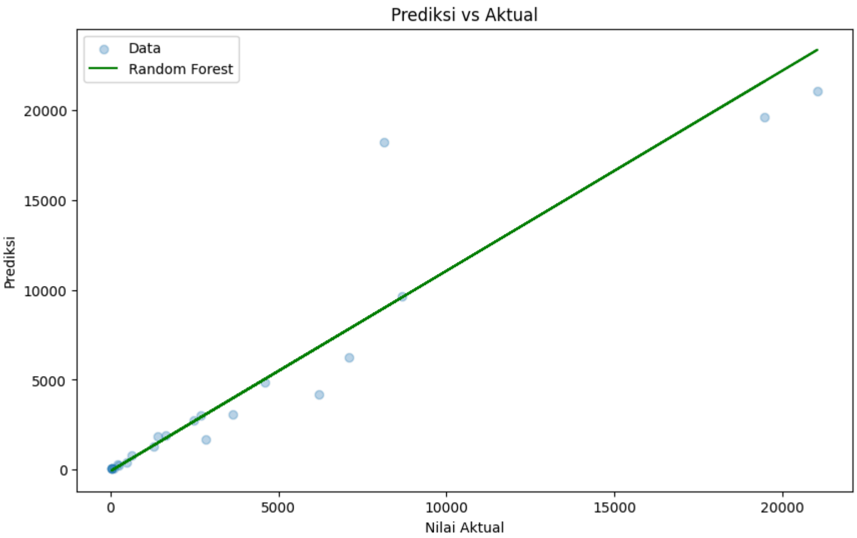
	Feature	Importance
0	Luas Panen	26.063501
1	Produktivitas	10.537301
2	Humidity	-4.340911

Next steps: Generate code with `lr_importance_df` View recommended plots

```
# 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:  [0.9806580432776951, 0.9838613986726765, 0.8937382726324672, 0.9899181247021213, 0.9390150025754918, 0.957
Rata-rata skor R-squared:  0.8254312994439771
Standar deviasi skor R-squared:  0.32482380975035197
```

```
# 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)
```

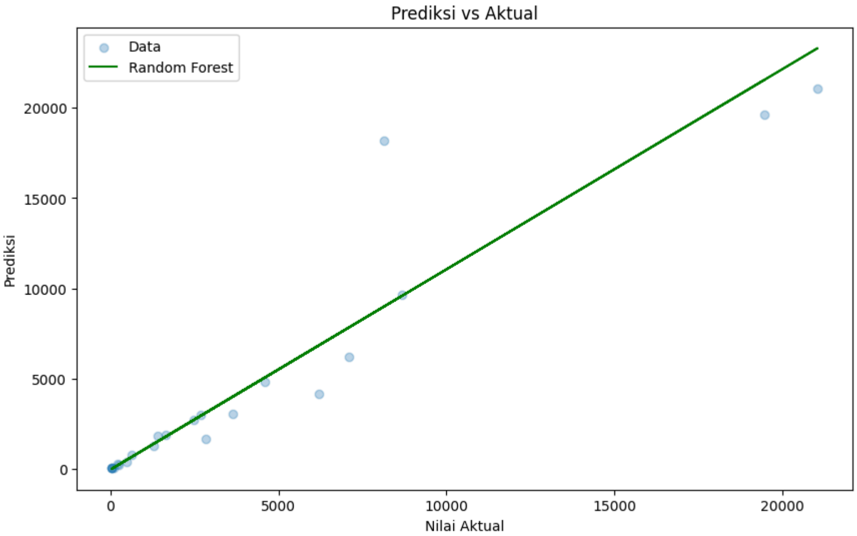
```
# K Fold SVR
kf_svr = KFold(n_splits=5, shuffle=True, random_state=42)

svr_r2_scores = []

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

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:  [0.989354049349939, 0.9847122821583368, 0.8831064335952922, 0.993980985341727, 0.9440720419878706, 0.95904
Rata-rata skor R-squared:  0.827506244824184
Standar deviasi skor R-squared:  0.3241457410688865
```

```
# 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 :  0.8431272882250188
RMSE :  2244.455078693873
MAE :  912.1437850430441
```

```
# 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)
```

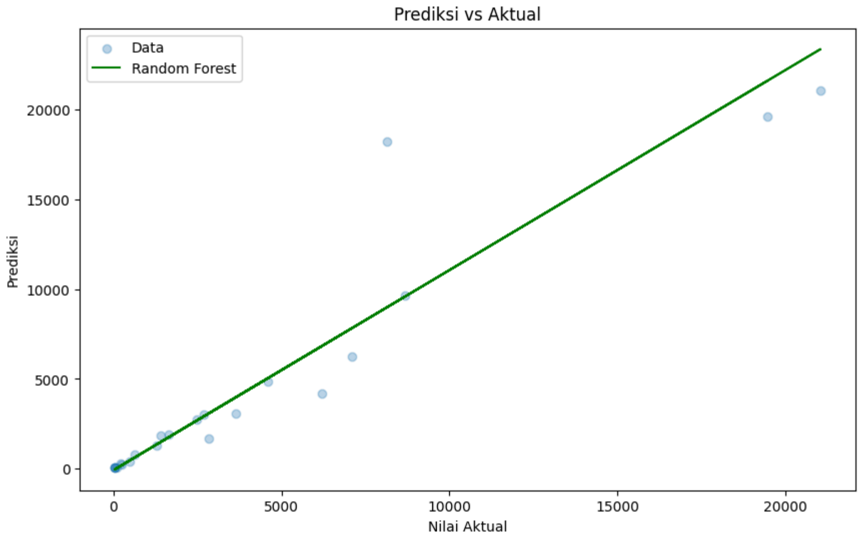
```
# Evaluation
print("R-Squared : ", r2_score(y_test, lasso_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, lasso_predict)))
print("MAE : ", mean_absolute_error(y_test, lasso_predict))
```

```
R-Squared :  0.8558295437732526
RMSE :  2151.6684512186334
MAE :  983.8210123011158
```

```
# 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:  [0.9806580435615322, 0.9838613987982083, 0.8937382730990062, 0.9899181247866096, 0.9390150033915445, 0.957
Rata-rata skor R-squared:  0.8254312997256209
Standar deviasi skor R-squared:  0.3248238099158103
```

```
# 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

```python
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)
```

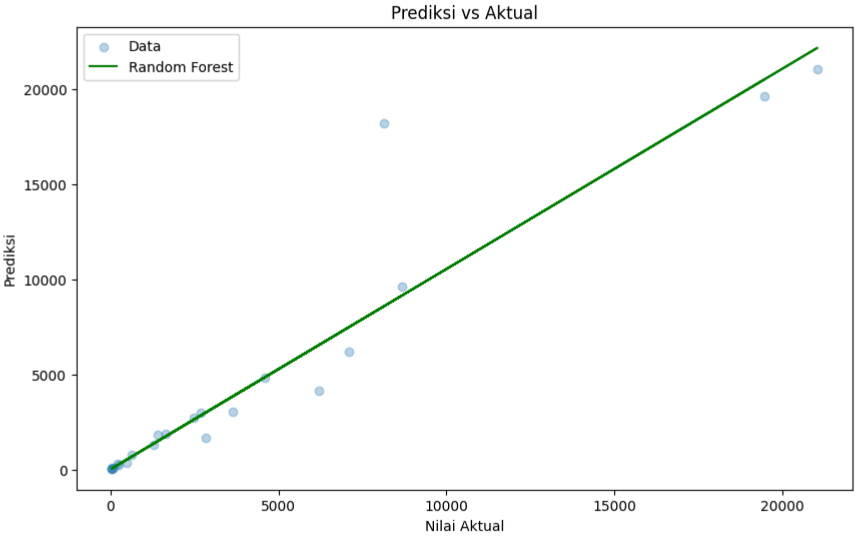
```python
# Evaluation
print("R-Squared : ", r2_score(y_test, xgb_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, xgb_predict)))
print("MAE : ", mean_absolute_error(y_test, xgb_predict))
```

```
R-Squared :  0.8656580954674586
RMSE :  2077.0309408911776
MAE :  753.849726234318
```

```python
# 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))
```

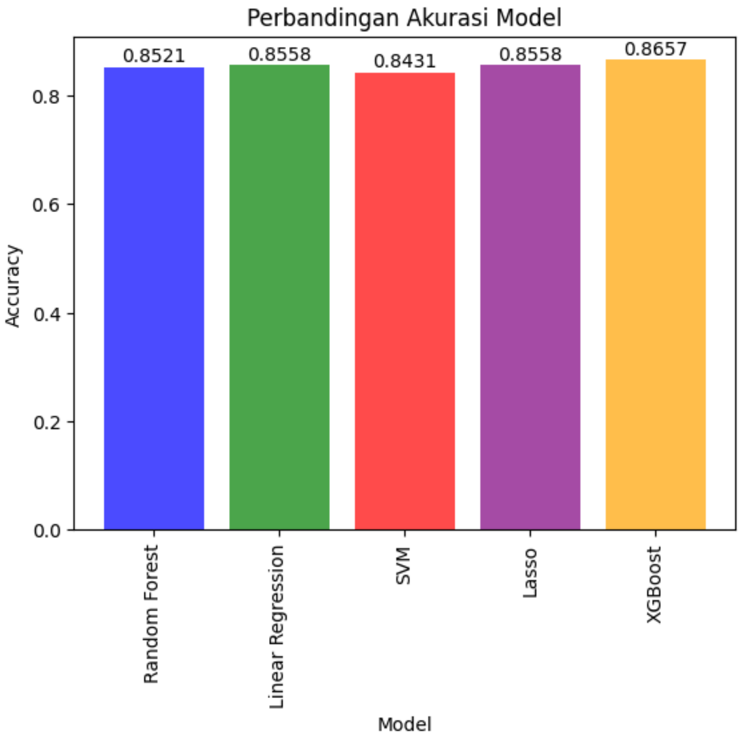
```
Skor untuk setiap fold:  [0.9979482974828257, 0.9884440318647028, 0.9146444368393949, 0.9941756960707039, 0.93284259625395, 0.96561
Rata-rata skor R-squared:  0.8322045351090598
Standar deviasi skor R-squared:  0.3280985141111987
```

```
# 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

```python
# Nama-nama model yang digunakan
model_names = ['Random Forest', 'Linear Regression', 'SVM', 'Lasso', 'XGBoost']

# Akurasi yang didapatkan dari masing-masing metode
accuracies = [
    r2_score(y_test, y_pred),
    r2_score(y_test, lr_predict),
    r2_score(y_test, svr_predict),
    r2_score(y_test, lasso_predict),
    r2_score(y_test, xgb_predict),
    # r2_score(y_test, ann_predict)
]

# Menetapkan posisi batang di sumbu X
x_pos = np.arange(len(model_names))

# Membuat diagram batang
plt.bar(x_pos, accuracies, color=['blue', 'green', 'red', 'purple', 'orange'], alpha=0.7)

# Menambahkan nilai pada setiap batang
for i in range(len(accuracies)):
    plt.text(x_pos[i], accuracies[i] + 0.01, f'{accuracies[i]:.4f}', ha='center')

# Menambahkan judul dan label
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.title('Perbandingan Akurasi Model')
plt.xticks(x_pos, model_names, rotation=90)  # Menetapkan nama model sebagai label sumbu X

# Menampilkan plot
plt.show()

r_sq_df = pd.DataFrame({
    'Model': model_names,
    'R-Squared': accuracies
})
r_sq_df
```



	Model	R-Squared
0	Random Forest	0.852059
1	Linear Regression	0.855830
2	SVM	0.843127
3	Lasso	0.855830
4	XGBoost	0.865658

Next steps: Generate code with `r_sq_df` View recommended plots