```
!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  84271      0 --:--:-- --:--:-- --:--:-- 84297
```

```
# 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
```

```
1 df = df.query("Tanaman == 'Ubi Kayu'")
```

```
1 df
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humid
362	Cilacap	2015	4789	87231.3	182.1493	Ubi Kayu	Meteorologi, Cilacap	8
363	Banyumas	2015	3134	51044.252	162.872534	Ubi Kayu	NaN	N
364	Purbalingga	2015	2773	74158.854	267.431857	Ubi Kayu	NaN	N
365	Banjarnegara	2015	7776	202053.878	259.84295	Ubi Kayu	NaN	N
366	Kebumen	2015	6807	167521.608	246.101966	Ubi Kayu	Sempor, Kebumen	7
...	...	...	...	...	...	...	...	
622	Semarang	2019	110,00	2374,00	215,84	Ubi Kayu	SMPK. Balit Getas, Semarang	8

Next steps: Generate code with df View recommended plots

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

```
<ipython-input-10-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-10-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 ', '')
```

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

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity

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

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity

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

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity

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

```
<ipython-input-14-95b6067be5a9>: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)
```

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

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         42
dtype: int64
```

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

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         42
dtype: int64
```

2. Pre-Processing Data

2.1 Mengubah Tipe Data Variabel Menjadi Integer

```
1 # Membuat function change data type
2 def clean_data(value):
3     if not isinstance(value, str):
4       if isinstance(value, int):
5         return float(value)
6       if isinstance(value, float):
7         return value
8       return np.nan
9     if value == '-':
10        return np.nan
11
12    rb = value.split(",", 1)
13    if len(rb) > 1:
14        value = rb[0] + "." + rb[1]
15    elif len(rb) == 1:
16        value = rb[0]
17    return float(value.replace(',', '.').replace(' ', '').strip())
```

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

```
<ipython-input-18-1c2a5001d4df>: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-18-1c2a5001d4df>: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-18-1c2a5001d4df>: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)
```

```
1 df.head()
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Humidity
362	Cilacap	2015	4789.0	87231.300	182.149300	Ubi Kayu	83.0
363	Banyumas	2015	3134.0	51044.252	162.872534	Ubi Kayu	NaN
364	Purbalingga	2015	2773.0	74158.854	267.431857	Ubi Kayu	NaN

Next steps: Generate code with df View recommended plots

2.2 Interpolate Data NaN

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

```
Kabupaten        0
Tahun            0
Luas Panen       5
Hasil Panen      5
Produktivitas    5
Tanaman          0
Humidity        42
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)
```

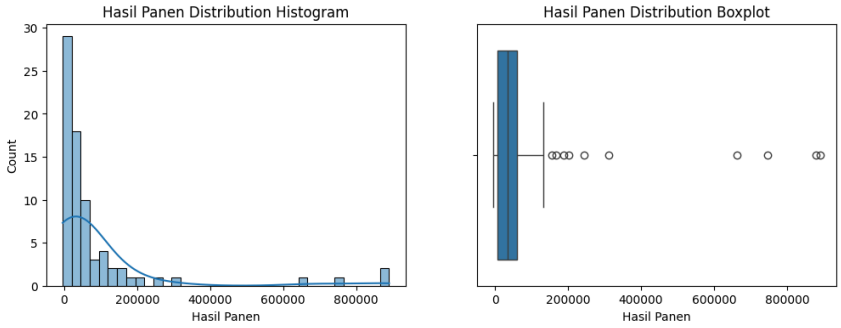
```
/usr/local/lib/python3.10/dist-packages/scipy/interpolate/_fitpack2.py:313: UserWarning:
The maximal number of iterations maxit (set to 20 by the program)
allowed for finding a smoothing spline with fp=s has been reached: s
too small.
There is an approximation returned but the corresponding weighted sum
of squared residuals does not satisfy the condition abs(fp-s)/s < tol.
  warnings.warn(message)
<ipython-input-21-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-21-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-21-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-21-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()
```

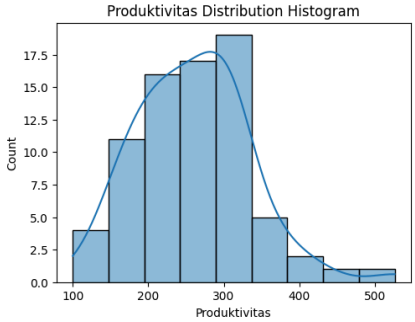
```
Kabupaten        0
Tahun            0
Luas Panen       0
Hasil Panen      0
Produktivitas    0
Tanaman          0
Humidity         0
dtype: int64
```

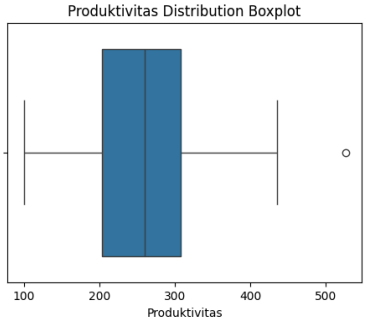
2.3 Handling outlier

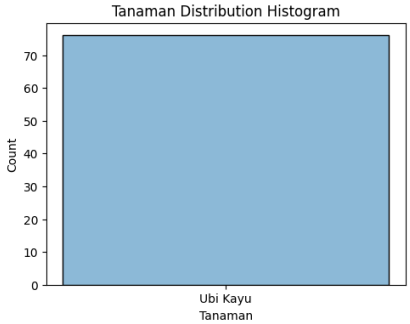
```
# Fungsi untuk menampilkan boxplot dan histogram
def num_dist(data, var):
    fig, ax = plt.subplots(1, 2, figsize=(12, 4))

    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)
```

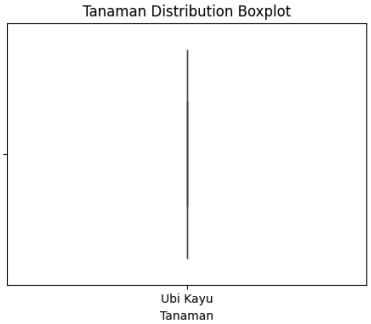

Hasil Panen Distribution Histogram
Count
Hasil Panen

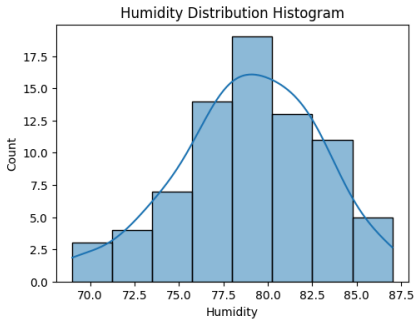
Hasil Panen Distribution Boxplot
Hasil Panen

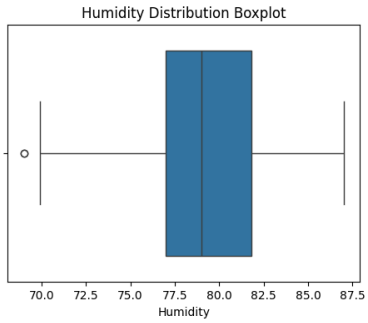

Produktivitas Distribution Histogram
Count
Produktivitas


Produktivitas Distribution Boxplot
Produktivitas


Tanaman Distribution Histogram
Count
Ubi Kayu
Tanaman

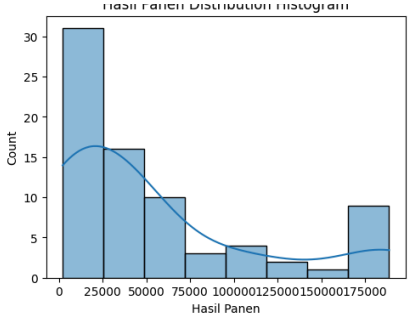

Tanaman Distribution Boxplot
Ubi Kayu
Tanaman

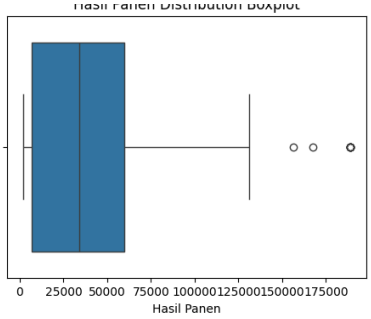

Humidity Distribution Histogram
Count
Humidity

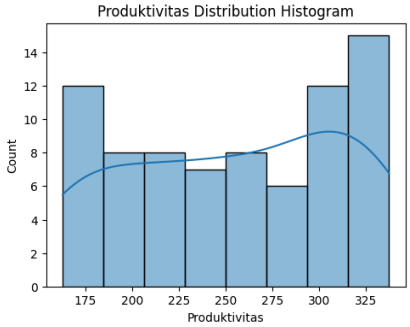

Humidity Distribution Boxplot
Humidity

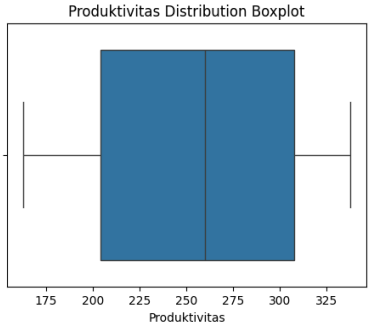
```
# Variabel yang terdapat outlier
outlier_var = ['Humidity', 'Produktivitas', 'Hasil Panen', 'Luas Panen']
threshold = 0.1

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

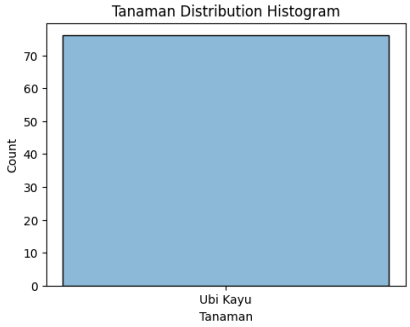
```
df_var = df.columns
for var in df_var:
    num_dist(df, var)
```

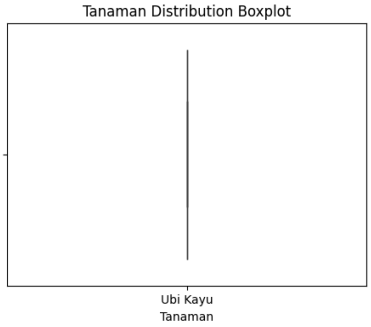

Count
Hasil Panen

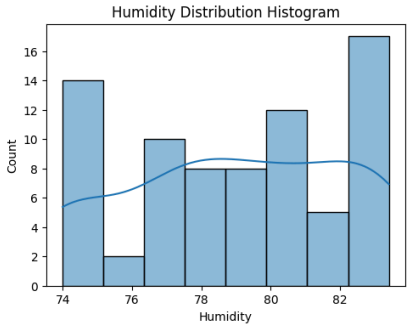

Hasil Panen

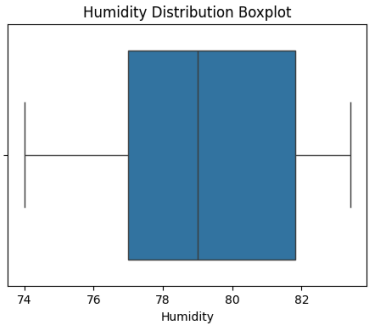

Produktivitas Distribution Histogram
Count
Produktivitas


Produktivitas Distribution Boxplot
Produktivitas


Tanaman Distribution Histogram
Count
Ubi Kayu
Tanaman


Tanaman Distribution Boxplot
Ubi Kayu
Tanaman


Humidity Distribution Histogram
Count
Humidity


Humidity Distribution Boxplot
Humidity

```
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%
Luas Panen	76.0	2065.055789	2191.724483	110.000000	362.250000	1443.500000
Hasil Panen	76.0	53603.583198	59661.909522	2182.000000	6608.366606	33924.518000
Produktivitas	76.0	255.370992	58.411112	162.872534	203.993103	259.877454
Humidity	76.0	78.986147	3.162214	74.000000	77.000000	79.000000

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-27-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-27-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

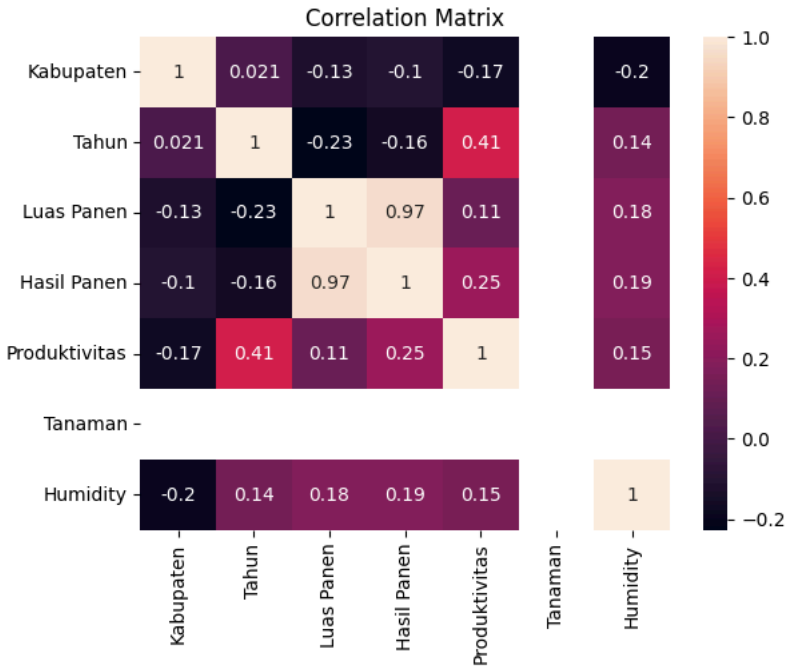
2.1 Correlation

```
df.corr()
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Humidi
Kabupaten	1.000000	0.020717	-0.128527	-0.104121	-0.171030	NaN	-0.1959
Tahun	0.020717	1.000000	-0.226570	-0.157323	0.412091	NaN	0.1412
Luas Panen	-0.128527	-0.226570	1.000000	0.965757	0.108288	NaN	0.1814
Hasil Panen	-0.104121	-0.157323	0.965757	1.000000	0.254952	NaN	0.1850
Produktivitas	-0.171030	0.412091	0.108288	0.254952	1.000000	NaN	0.1543
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 : (60, 3)
x_test : (16, 3)
y_train : (60,)
y_test : (16,)
```

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)
```

```
0.980150217019577
```

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

```
Luas Panen       0.940233
Produktivitas    0.048442
Humidity         0.011326
dtype: float64
```

```
# 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	0.940233
1	Produktivitas	0.048442
2	Humidity	0.011326

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

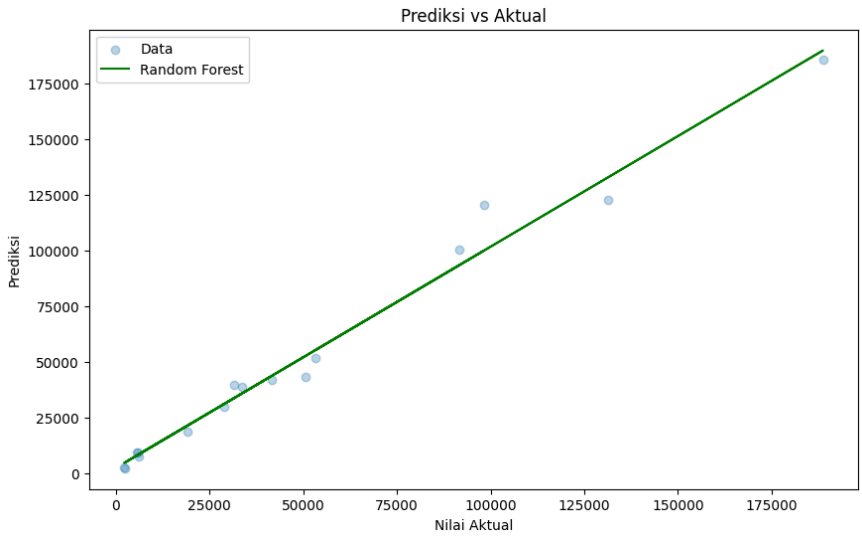
```
# K Fold RF
kf_rf = KFold(n_splits=5, shuffle=True, random_state=42)

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:  [0.8891966900459578, 0.8310324783491831, 0.973165262973765, 0.9276650231330473, 0.5141284194189147, 0.8270
Rata-rata skor R-squared:  0.7301802599345198
Standar deviasi skor R-squared:  0.2744572033950519
```

```
# 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 :  0.980150217019577
RMSE :  7250.490053200414
MAE :  4742.571177794228
```

```
# 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 :  0.982947091420275
RMSE :  6720.300785375411
MAE :  4503.71407464117
```

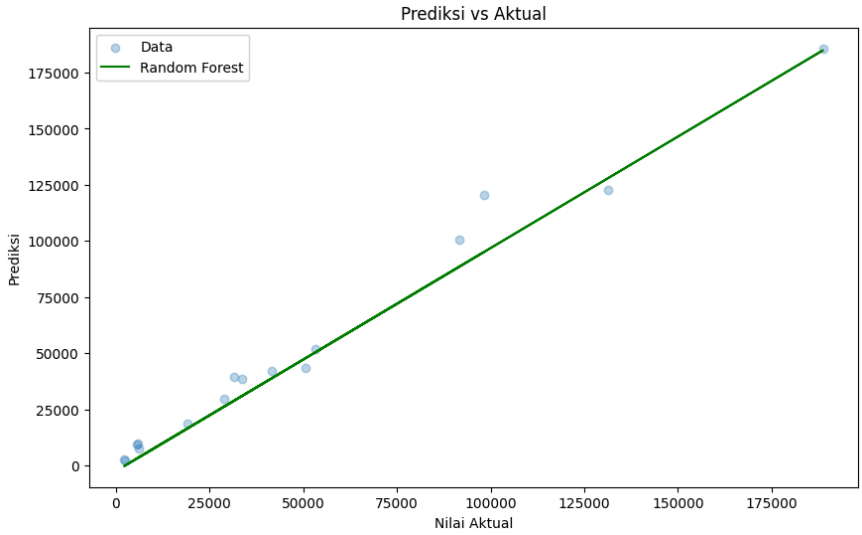
	Feature	Importance
0	Luas Panen	25.920061
1	Produktivitas	164.766486
2	Humidity	-292.044171

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:  [0.8968002903391974, 0.8848174162482655, 0.9804873499356268, 0.9406739746117769, 0.9576111854326027, 0.932(
Rata-rata skor R-squared:  0.8036431468779208
Standar deviasi skor R-squared:  0.31608302011067485
```

```
# 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)
```

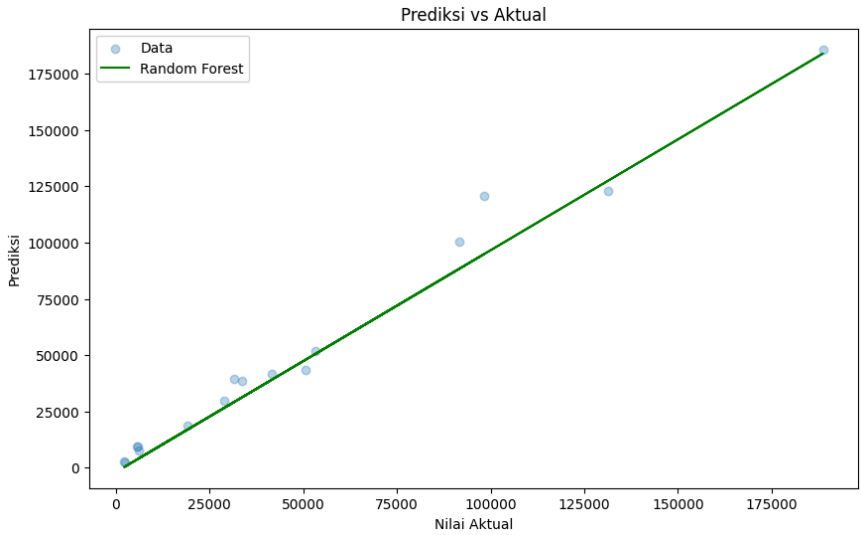
```
# K Fold SVR
kf_svr = KFold(n_splits=5, shuffle=True, random_state=42)

svr_r2_scores = []

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

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:  [0.8897664787923816, 0.8839722604022386, 0.9794915188519879, 0.9592776829261592, 0.9227871276547607, 0.927
Rata-rata skor R-squared:  0.7995198318834635
Standar deviasi skor R-squared:  0.3140142996585972
```

```
# 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 :  0.9852312011679489
RMSE :  6254.060340376949
MAE :  4552.993391856489
```

```
# 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)
```

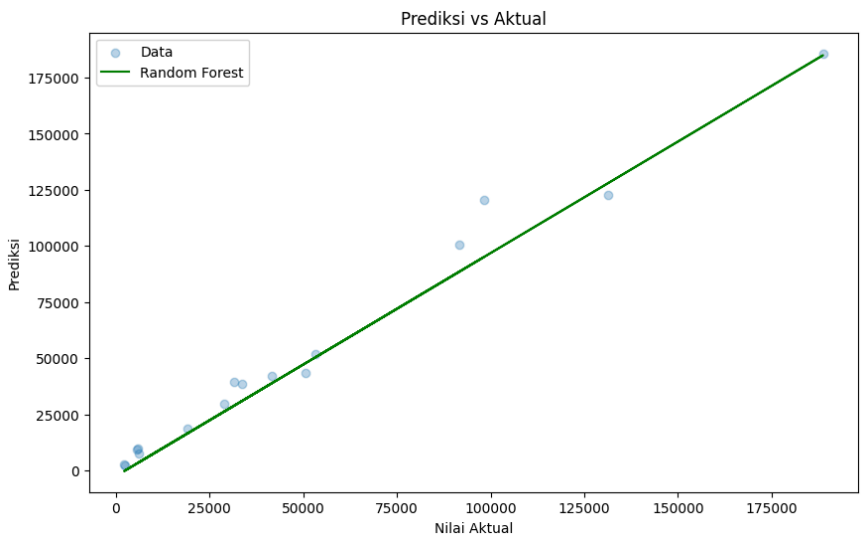
```
# Evaluation
print("R-Squared : ", r2_score(y_test, lasso_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, lasso_predict)))
print("MAE : ", mean_absolute_error(y_test, lasso_predict))
```

```
R-Squared :  0.9829470914561323
RMSE :  6720.300778309992
MAE :  4503.71406336921
```

```
# 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:  [0.8968002903613919, 0.8848174162585621, 0.9804873499929577, 0.9406739747104339, 0.9576111855409872, 0.932
Rata-rata skor R-squared:  0.8036431469325029
Standar deviasi skor R-squared:  0.31608302012466905
```

```
# 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)
```

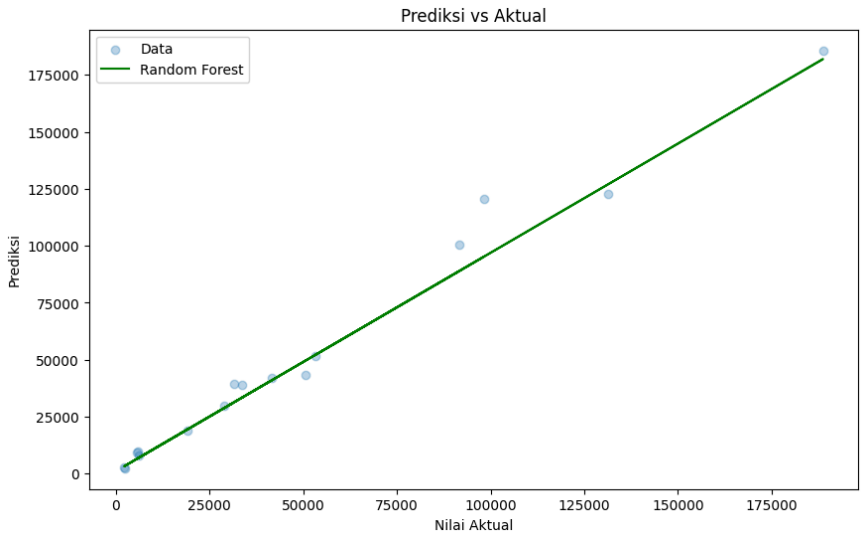
```
# Evaluation
print("R-Squared : ", r2_score(y_test, xgb_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, xgb_predict)))
print("MAE : ", mean_absolute_error(y_test, xgb_predict))
```

```
R-Squared :  0.9760841438443281
RMSE :  7958.522454212718
MAE :  4769.016467937027
```

```
# 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))
```

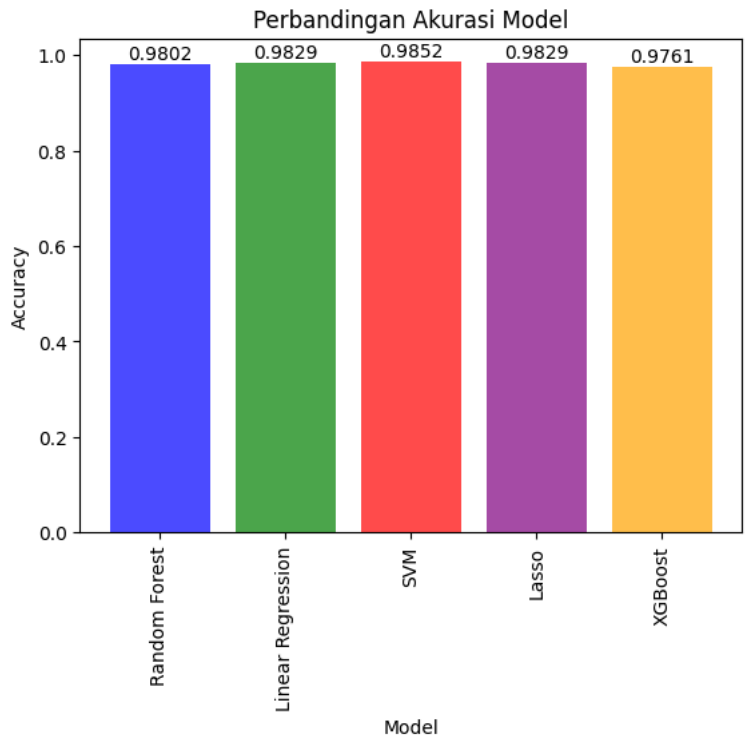
```
Skor untuk setiap fold:  [0.8675457292077473, 0.8103426785799467, 0.9572562739451295, 0.8954374810143046, 0.6667351351655685, 0.839
Rata-rata skor R-squared:  0.7323813610929214
Standar deviasi skor R-squared:  0.275182051509695
```

```
# 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

```
# 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	0.980150
1	Linear Regression	0.982947
2	SVM	0.985231
3	Lasso	0.982947
4	XGBoost	0.976084

Next steps: Generate code with r_sq_df View recommended plots