

```
1 !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)

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.model_selection import GridSearchCV
6 from sklearn.preprocessing import MinMaxScaler
7 from scipy.stats.mstats import winsorize
8 from sklearn.preprocessing import LabelEncoder
9 from sklearn.model_selection import KFold
10 from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, mean_absolute_percentage_error
11
```

1. Data Understanding

```
1 !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  103k      0 --:--:-- --:--:-- --:--:-- 103k

1 # Path ke file Excel
2 file_path = '/content/FINAL_DATASET_with_Humidity_and_Station.xlsx'
3
4 df = pd.read_excel(file_path)
```

```
1 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

```
1 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 == 'Kacang Hijau'")
```


```
1 df
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity
325	Cilacap	2015	796	1076.021	13.517852	Kacang Hijau	Meteorologi, Cilacap	83.0
326	Banyumas	2015	187	92	4.919786	Kacang Hijau	NaN	NaN
327	Purbalingga	2015	7	9.556	13.651429	Kacang Hijau	NaN	NaN
328	Banjarnegara	2015	-	-	-	Kacang Hijau	NaN	NaN
329	Kebumen	2015	10439	13619	13.046269	Kacang Hijau	Sempor, Kebumen	77.0
...	...	...	...	...	...	...	...	...
583	Semarang	2019	-	-	-	Kacang Hijau	SMPK. Balit Getas, Semarang	81.0

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-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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
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-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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
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-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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
df['Produktivitas'] = df['Produktivitas'].apply(clean\_data)

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

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Humidity
325	Cilacap	2015	796.0	1076.021	13.517852	Kacang Hijau	83.0
326	Banyumas	2015	187.0	92.000	4.919786	Kacang Hijau	NaN
327	Purbalingga	2015	7.0	9.556	13.651429	Kacang Hijau	NaN

Next steps: [Generate code with df](#) [View recommended plots](#)

### 2.2 Interpolate Data NaN

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

```

Kabupaten      0
Tahun          0
Luas Panen     30
Hasil Panen    30
Produktivitas  30
Tanaman        0
Humidity       42
dtype: int64

```

```

1 df['Luas Panen'] = df['Luas Panen'].interpolate(method='spline', order=2)
2 df['Hasil Panen'] = df['Hasil Panen'].interpolate(method='spline', order=2)
3 df['Produktivitas'] = df['Produktivitas'].interpolate(method='spline', order=2)
4 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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
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-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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

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

```

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

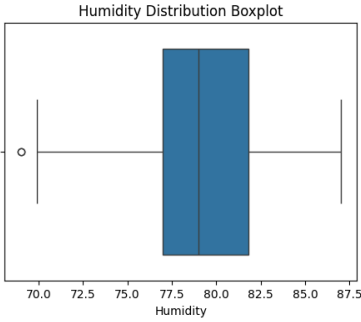
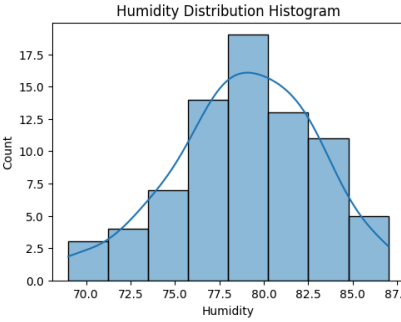
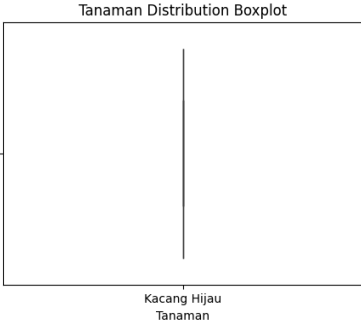
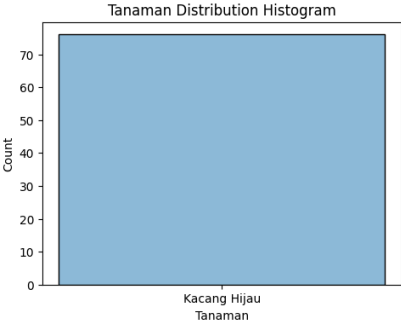
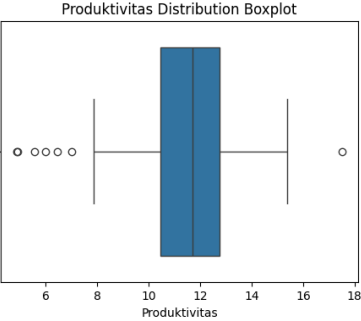
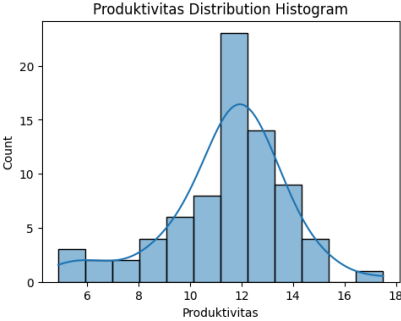
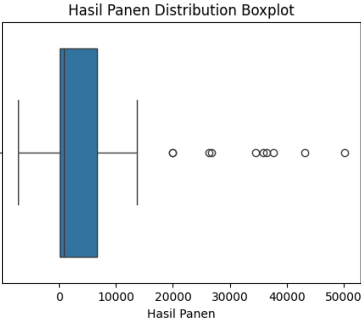
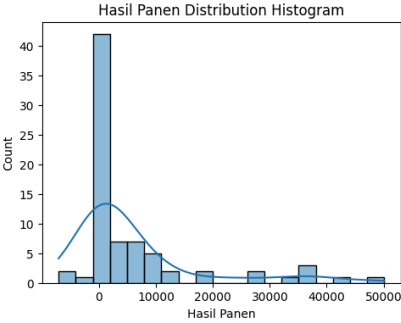
```

## 2.3 Handling outlier

```

1 # Fungsi untuk menampilkan boxplot dan histogram
2 def num_dist(data, var):
3     fig, ax = plt.subplots(1, 2, figsize=(12, 4))
4
5     sns.histplot(data=data, x=var, kde=True, ax=ax[0])
6     sns.boxplot(data=data, x=var, ax=ax[1])
7     ax[0].set_title(f"{var} Distribution Histogram")
8     ax[1].set_title(f"{var} Distribution Boxplot")
9
10    plt.show()
11
12 df_var = df.columns
13 for var in df_var:
14     num_dist(df, var)

```

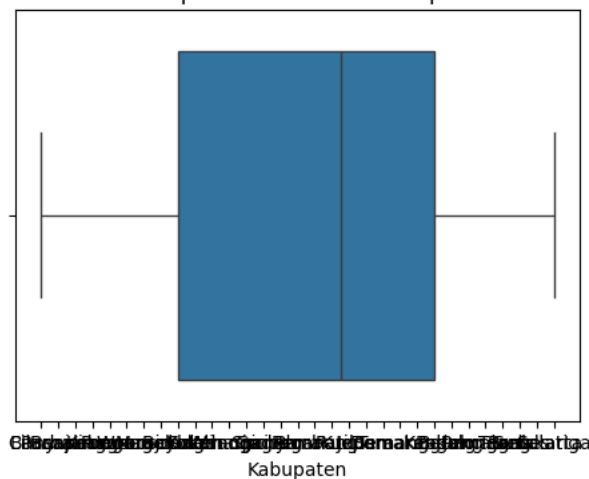


```
1 # Variabel yang terdapat outlier
2 outlier_var = ['Humidity', 'Produktivitas', 'Hasil Panen', 'Luas Panen']
3 threshold = 0.1
4
5 # Menggunakan treatment Winsorize untuk menghapus Outlier
6 for var in outlier_var:
7     df.loc[:, var] = winsorize(df[var], limits=[threshold, threshold])

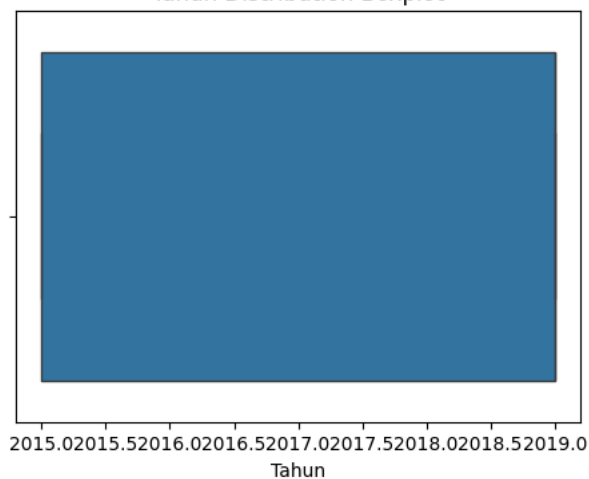
1 df_var = df.columns
2 for var in df_var:
3     num_dist(df, var)
```



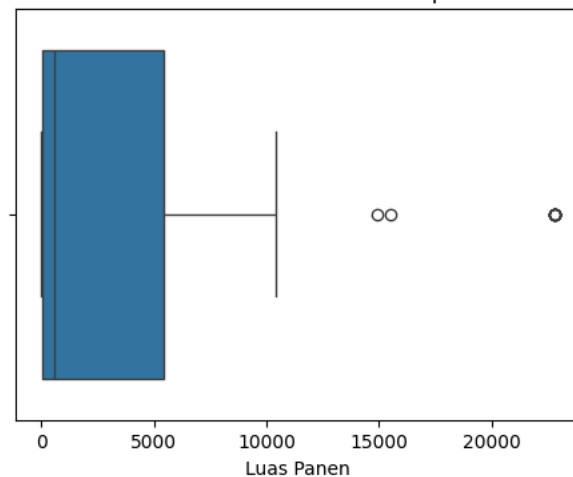
### Kabupaten Distribution Boxplot



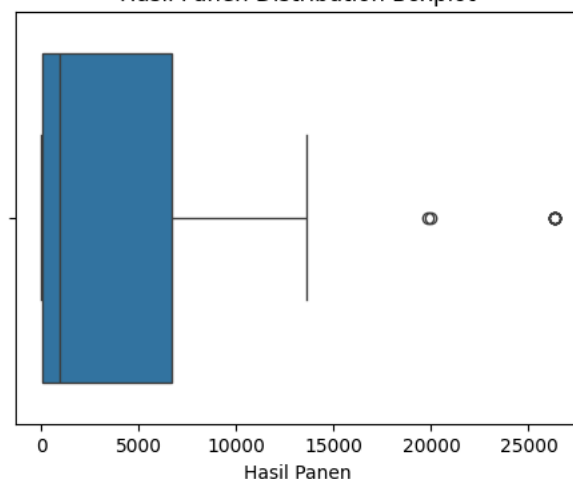
### Tahun Distribution Boxplot

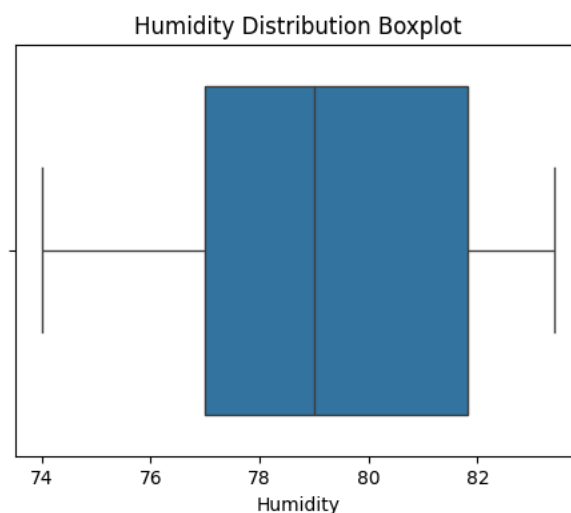
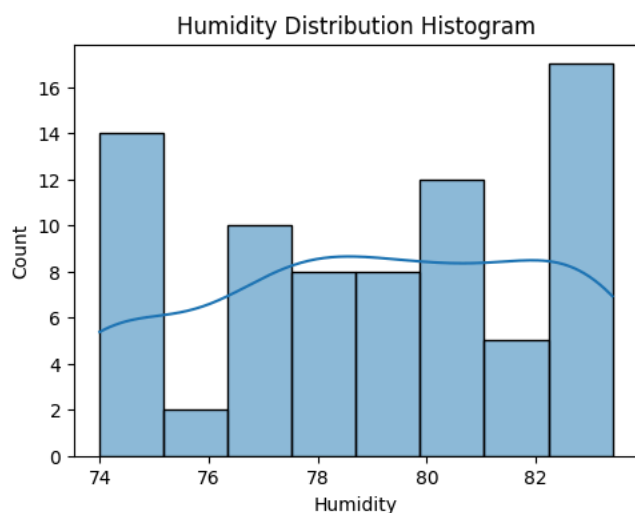
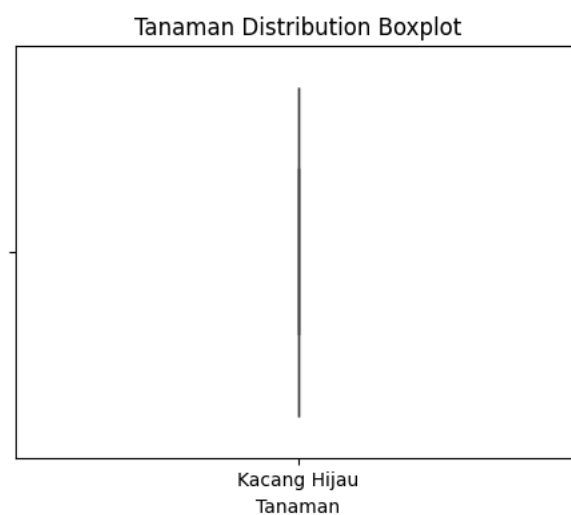
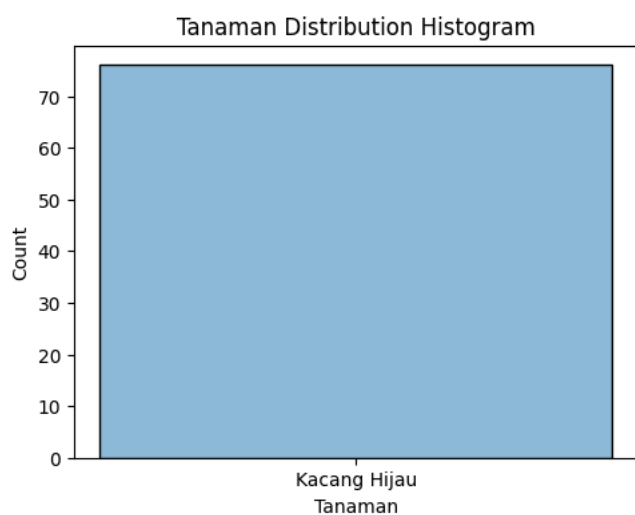
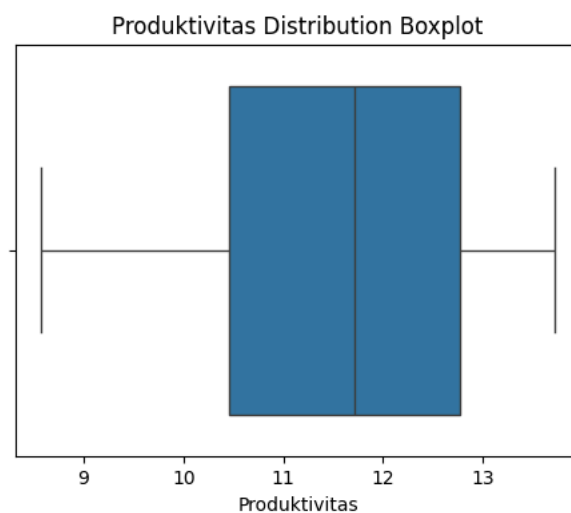
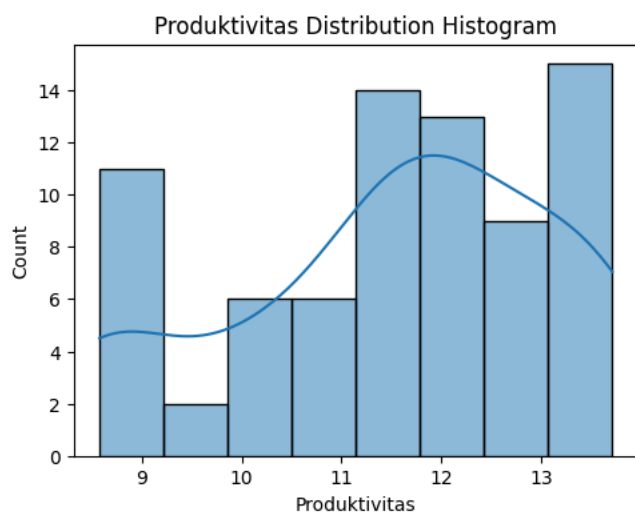


### Luas Panen Distribution Boxplot



Hasil Panen Distribution Boxplot





```

1 numerical_columns = ['Luas Panen', 'Hasil Panen', 'Produktivitas', 'Humidity']
2
3
4 description = df[numerical_columns].describe().T
5
6 # Menambahkan percentiles ke deskripsi
7 description['25%'] = df[numerical_columns].quantile(0.25)
8 description['50%'] = df[numerical_columns].quantile(0.50)
9 description['75%'] = df[numerical_columns].quantile(0.75)
10
11 # Menyusun ulang kolom agar sesuai dengan format tabel di gambar
12 description = description[['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max']]
13 description

```



	count	mean	std	min	25%	50%	75%	max
<b>Luas Panen</b>	76.0	4458.866886	7205.058102	1.000000	23.500000	592.977407	5432.083601	22790.000000
<b>Hasil Panen</b>	76.0	5298.326156	8428.683740	2.000000	41.280000	927.256000	6682.500000	26317.000000
<b>Produktivitas</b>	76.0	11.534875	1.620113	8.578571	10.460431	11.717258	12.762194	13.712508
<b>Humidity</b>	76.0	78.986149	3.162229	74.000000	77.000000	79.000000	81.830680	83.407512

Next steps:

[Generate code with description](#)[View recommended plots](#)

## 2.4 Encoding

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

<ipython-input-26-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](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)

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

<ipython-input-26-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](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

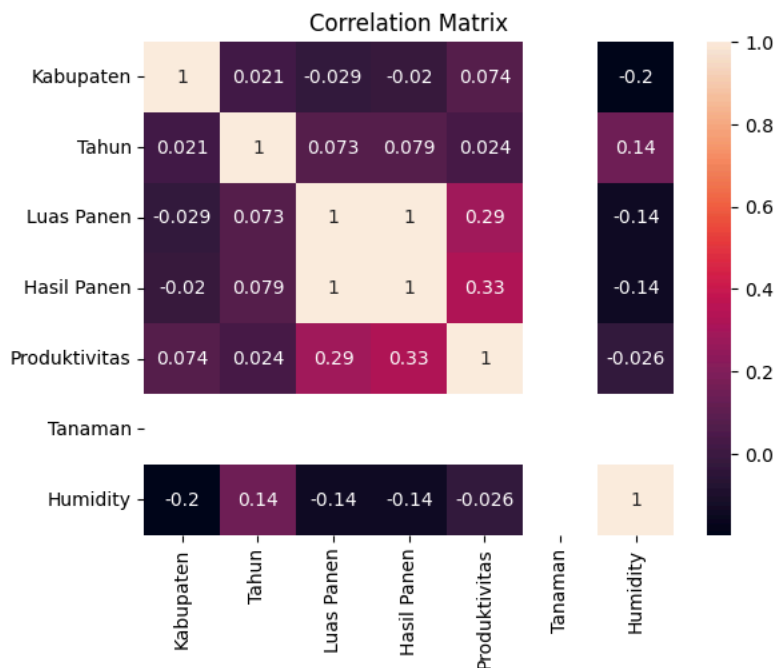
### 2.5 Correlation

```
1 df.corr()
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Humidity
<b>Kabupaten</b>	1.000000	0.020717	-0.029023	-0.020095	0.073651	NaN	-0.195902
<b>Tahun</b>	0.020717	1.000000	0.073340	0.079341	0.024028	NaN	0.141254
<b>Luas Panen</b>	-0.029023	0.073340	1.000000	0.995439	0.293560	NaN	-0.143836
<b>Hasil Panen</b>	-0.020095	0.079341	0.995439	1.000000	0.327403	NaN	-0.144753
<b>Produktivitas</b>	0.073651	0.024028	0.293560	0.327403	1.000000	NaN	-0.025725
<b>Tanaman</b>	NaN	NaN	NaN	NaN	NaN	NaN	NaN
<b>Humidity</b>	-0.195902	0.141254	-0.143836	-0.144753	-0.025725	NaN	1.000000

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

Text(0.5, 1.0, 'Correlation Matrix')



### 3. Modelling

```
1 from sklearn.model_selection import train_test_split
2
3 x = df.drop(["Hasil Panen", "Kabupaten", "Tahun", "Tanaman"], axis=1)
4 y = df["Hasil Panen"]
5
6 # Splitting data set - 25% test dataset and 75%
7
8 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=5)
9
10 print("x_train :",x_train.shape)
11 print("x_test :",x_test.shape)
12 print("y_train :",y_train.shape)
13 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

```
1 from sklearn.ensemble import RandomForestRegressor
2
3 rf_model = RandomForestRegressor(n_estimators = 11)
4 rf_model.fit(x_train,y_train)
5 rf_predict = rf_model.predict(x_test)
```

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

```
0.9932992008069733
```

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

```
Luas Panen      0.979929
Produktivitas   0.010048
Humidity        0.010023
dtype: float64
```

```

1 # Get feature importances
2 feature_importances = rf_model.feature_importances_
3 features = x.columns
4
5 # Create a DataFrame for better visualization
6 feature_importance_df = pd.DataFrame({
7     'Feature': features,
8     'Importance': feature_importances
9 })
10
11 # Sort the DataFrame by importance
12 feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
13
14 # Display the feature importances DataFrame
15 feature_importance_df

```



	Feature	Importance	
0	Luas Panen	0.979929	
1	Produktivitas	0.010048	
2	Humidity	0.010023	



Next steps:

[Generate code with feature\\_importance\\_df](#)[View recommended plots](#)

```

1 # K Fold RF
2 kf_rf = KFold(n_splits=5, shuffle=True, random_state=42)
3
4 rf_r2_scores = []
5
6 rf_model_kf = RandomForestRegressor(n_estimators = 11)
7
8 for train_index, test_index in kf_rf.split(x):
9     x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
10    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]
11
12    # Melatih model
13    rf_model_kf.fit(x_train_fold, y_train_fold)
14
15    # Memprediksi hasil pada data uji
16    y_pred_fold = rf_model_kf.predict(x_test_fold)
17
18    # Menghitung skor R-squared
19    r2 = r2_score(y_test_fold, y_pred_fold)
20    rf_r2_scores.append(r2)
21
22 rf_r2_scores.append(np.mean(rf_r2_scores))
23 rf_r2_scores.append(np.std(rf_r2_scores))
24 # Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
25 print("Skor untuk setiap fold: ", rf_r2_scores)
26 print("Rata-rata skor R-squared: ", np.mean(rf_r2_scores))
27 print("Standar deviasi skor R-squared: ", np.std(rf_r2_scores))

```



```

Skor untuk setiap fold: [0.9781045997255579, 0.9928072606365212, 0.9729427677612363, 0.9572041023305491, 0.980934957649809, 0.97635
Rata-rata skor R-squared: 0.8384263036563535
Standar deviasi skor R-squared: 0.3381042930954794

```

```

1 # Evaluation
2 y_pred = rf_model.predict(x_test)
3
4 print("R-Squared : ", r2_score(y_test, y_pred))
5 print("RMSE : ", np.sqrt(mean_squared_error(y_test, y_pred)))
6 print("MAE : ", mean_absolute_error(y_test, y_pred))

```



```

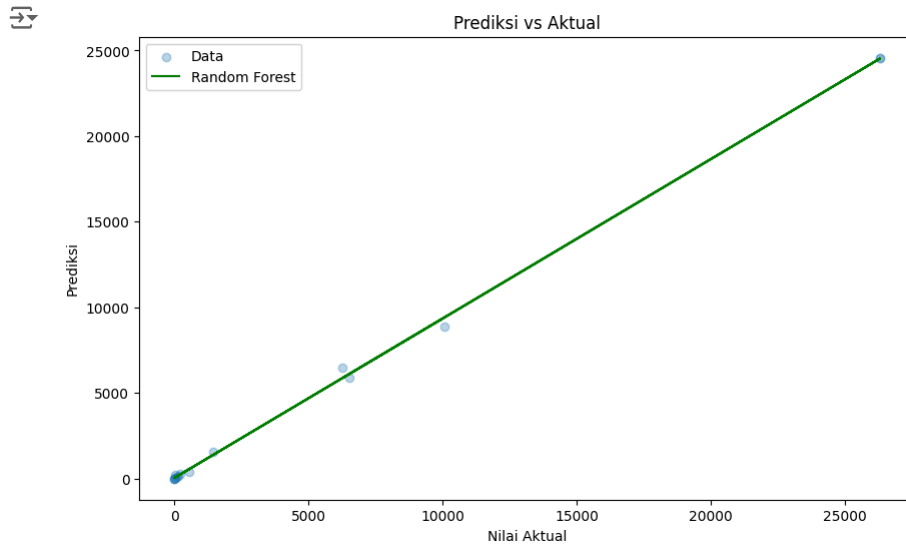
R-Squared : 0.9932992008069733
RMSE : 706.0928962739378
MAE : 378.8546997282248

```

```

1 # Membuat plot
2 plt.figure(figsize=(10, 6))
3 plt.scatter(y_test, y_pred, alpha=0.3, label='Data')
4
5 # Menghitung garis regresi linear
6 coef = np.polyfit(y_test, y_pred, 1)
7 poly1d_fn = np.poly1d(coef)
8
9 # Menambahkan garis regresi linear pada plot
10 plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')
11
12 plt.xlabel('Nilai Aktual')
13 plt.ylabel('Prediksi')
14 plt.title('Prediksi vs Aktual')
15 plt.legend()
16 plt.show()

```






## 3.2 Linear Regression

```

1 from sklearn.linear_model import LinearRegression
2
3 # Inisialisasi model Linear Regression
4 lr_model = LinearRegression()
5
6 # Melatih model
7 lr_model.fit(x_train, y_train)
8
9 # Memprediksi hasil pada data uji
10 lr_predict = lr_model.predict(x_test)
11
12 # Evaluation
13 print("R-Squared : ", r2_score(y_test, lr_predict))
14 print("RMSE : ", np.sqrt(mean_squared_error(y_test, lr_predict)))
15 print("MAE : ", mean_absolute_error(y_test, lr_predict))
16
17 # Mengambil koefisien sebagai feature importance
18 lr_importance = lr_model.coef_
19
20 # Membuat dataframe untuk feature importance
21 lr_importance_df = pd.DataFrame({
22     'Feature': x_train.columns,
23     'Importance': lr_importance
24 })
25
26 lr_importance_df

```

 R-Squared : 0.9969546235223951  
 RMSE : 476.0132427709749  
 MAE : 422.530666715847


	Feature	Importance	
0	Luas Panen	1.153775	
1	Produktivitas	234.047371	
2	Humidity	-5.791519	

Next steps: [Generate code with lr\\_importance\\_df](#) [View recommended plots](#)

```

1 # K Fold LR
2 kf_lr = KFold(n_splits=5, shuffle=True, random_state=42)
3
4 lr_r2_scores = []
5
6 lr_model_kf = LinearRegression()
7
8 for train_index, test_index in kf_lr.split(x):
9     x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
10    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]
11
12    # Melatih model
13    lr_model_kf.fit(x_train_fold, y_train_fold)
14
15    # Memprediksi hasil pada data uji
16    y_pred_fold = lr_model_kf.predict(x_test_fold)
17
18    # Menghitung skor R-squared
19    r2 = r2_score(y_test_fold, y_pred_fold)
20    lr_r2_scores.append(r2)
21
22 lr_r2_scores.append(np.mean(lr_r2_scores))
23 lr_r2_scores.append(np.std(lr_r2_scores))
24 # Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
25 print("Skor untuk setiap fold: ", lr_r2_scores)
26 print("Rata-rata skor R-squared: ", np.mean(lr_r2_scores))
27 print("Standar deviasi skor R-squared: ", np.std(lr_r2_scores))

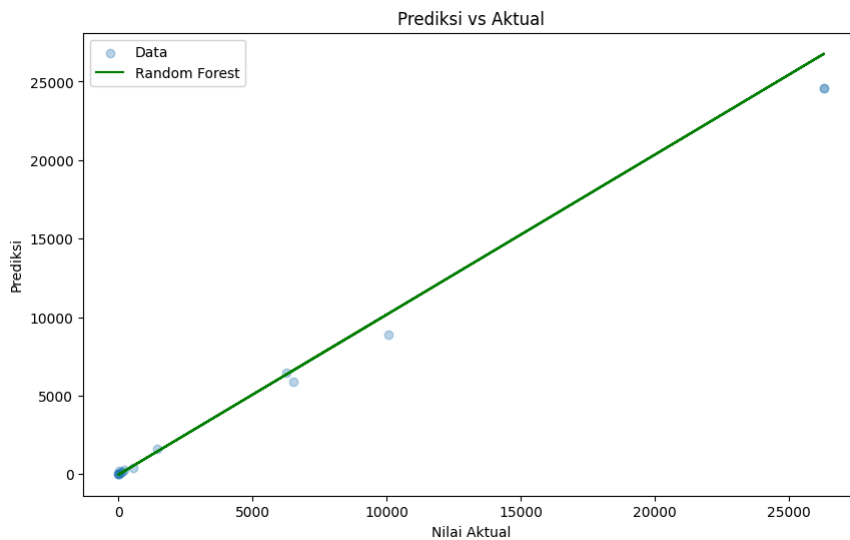
```

 Skor untuk setiap fold: [0.995225210487803, 0.9958900510610218, 0.989850020828672, 0.9663325828901652, 0.9949223721582379, 0.988444  
 Rata-rata skor R-squared: 0.8487065800754844  
 Standar deviasi skor R-squared: 0.342417831849221

```

1 # Membuat plot
2 plt.figure(figsize=(10, 6))
3 plt.scatter(y_test, y_pred, alpha=0.3, label='Data')
4
5 # Menghitung garis regresi linear
6 coef = np.polyfit(y_test, lr_predict, 1)
7 poly1d_fn = np.poly1d(coef)
8
9 # Menambahkan garis regresi linear pada plot
10 plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')
11
12 plt.xlabel('Nilai Aktual')
13 plt.ylabel('Prediksi')
14 plt.title('Prediksi vs Aktual')
15 plt.legend()
16 plt.show()

```



### 3.3 Support Vector Machine

```
1 from sklearn.svm import SVR
2
3 # Inisialisasi model SVR
4 svr_model = SVR(C= 1000, degree=3, gamma='auto', kernel='linear') # Anda bisa menggunakan kernel lain seperti 'linear', 'poly', dll.
5
6 # Melatih model
7 svr_model.fit(x_train, y_train) # Menggunakan y_train.ravel() untuk mengubah bentuk jika diperlukan
8
9 # Memprediksi hasil pada data uji
10 svr_predict = svr_model.predict(x_test)
```

```
1 # K Fold SVR
2 kf_svr = KFold(n_splits=5, shuffle=True, random_state=42)
3
4 svr_r2_scores = []
5
6 svr_model_kf = SVR(C= 1000, degree=3, gamma='auto', kernel='linear') # Anda bisa menggunakan kernel lain seperti 'linear', 'poly', dll.
7
8 for train_index, test_index in kf_svr.split(x):
9     x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
10    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]
11
12    # Melatih model
13    svr_model_kf.fit(x_train_fold, y_train_fold)
14
15    # Memprediksi hasil pada data uji
16    y_pred_fold = svr_model_kf.predict(x_test_fold)
17
18    # Menghitung skor R-squared
19    r2 = r2_score(y_test_fold, y_pred_fold)
20    svr_r2_scores.append(r2)
21
22 svr_r2_scores.append(np.mean(svr_r2_scores))
23 svr_r2_scores.append(np.std(svr_r2_scores))
24 # Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
25 print("Skor untuk setiap fold: ", svr_r2_scores)
26 print("Rata-rata skor R-squared: ", np.mean(svr_r2_scores))
27 print("Standar deviasi skor R-squared: ", np.std(svr_r2_scores))
```



```
Skor untuk setiap fold: [0.9637329504778158, -0.6328210351144523, 0.674507755155004, 0.9126722644925767, 0.8280883384208625, 0.5492
Rata-rata skor R-squared: 0.5489024730737271
Standar deviasi skor R-squared: 0.5063325823254847
```

```

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

```

```

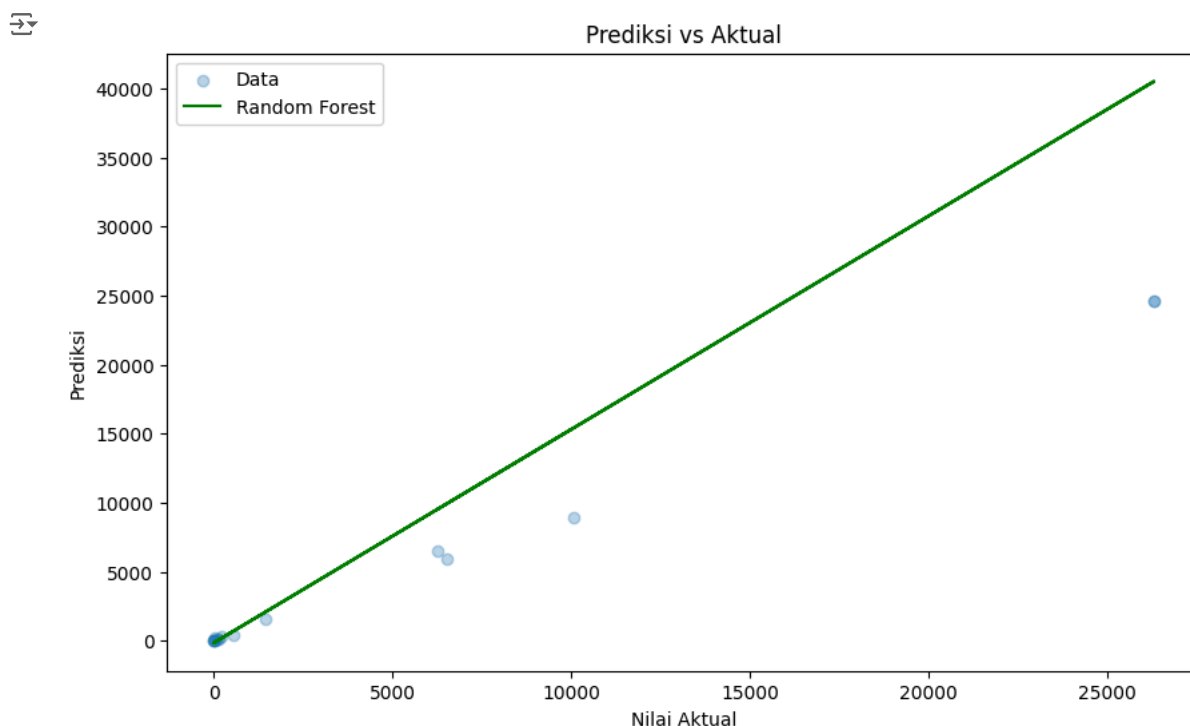
↗ R-Squared : 0.6135970431832414
  RMSE : 5361.9006297455035
  MAE : 2508.5760926248913

```

```

1 # Membuat plot
2 plt.figure(figsize=(10, 6))
3 plt.scatter(y_test, y_pred, alpha=0.3, label='Data')
4
5 # Menghitung garis regresi linear
6 coef = np.polyfit(y_test, svr_predict, 1)
7 poly1d_fn = np.poly1d(coef)
8
9 # Menambahkan garis regresi linear pada plot
10 plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')
11
12 plt.xlabel('Nilai Aktual')
13 plt.ylabel('Prediksi')
14 plt.title('Prediksi vs Aktual')
15 plt.legend()
16 plt.show()

```



### ✓ 3.4 Lasso Regression

```

1 from sklearn.linear_model import Lasso
2
3 # Inisialisasi model Lasso Regression
4 lasso_model = Lasso(alpha=0.0001) # Anda bisa mengatur nilai alpha sesuai kebutuhan
5
6 # Melatih model
7 lasso_model.fit(x_train, y_train)
8
9 # Memprediksi hasil pada data uji
10 lasso_predict = lasso_model.predict(x_test)

```

```

1 # Evaluation
2
3 print("R-Squared : ", r2_score(y_test, lasso_predict))
4 print("RMSE : ", np.sqrt(mean_squared_error(y_test, lasso_predict)))
5 print("MAE : ", mean_absolute_error(y_test, lasso_predict))

```

```

↗ R-Squared : 0.996954623939775
  RMSE : 476.01321015130304
  MAE : 422.53062509788015

```

```

1 # K Fold Lasso
2 kf_lasso = KFold(n_splits=5, shuffle=True, random_state=42)
3
4 lasso_r2_scores = []
5
6 lasso_model_kf = Lasso(alpha=0.0001)
7
8 for train_index, test_index in kf_lasso.split(x):
9     x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
10    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]
11
12    # Melatih model
13    lasso_model_kf.fit(x_train_fold, y_train_fold)
14
15    # Memprediksi hasil pada data uji
16    y_pred_fold = lasso_model_kf.predict(x_test_fold)
17
18    # Menghitung skor R-squared
19    r2 = r2_score(y_test_fold, y_pred_fold)
20    lasso_r2_scores.append(r2)
21
22 lasso_r2_scores.append(np.mean(lasso_r2_scores))
23 lasso_r2_scores.append(np.std(lasso_r2_scores))
24 # Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
25 print("Skor untuk setiap fold: ", lasso_r2_scores)
26 print("Rata-rata skor R-squared: ", np.mean(lasso_r2_scores))
27 print("Standar deviasi skor R-squared: ", np.std(lasso_r2_scores))

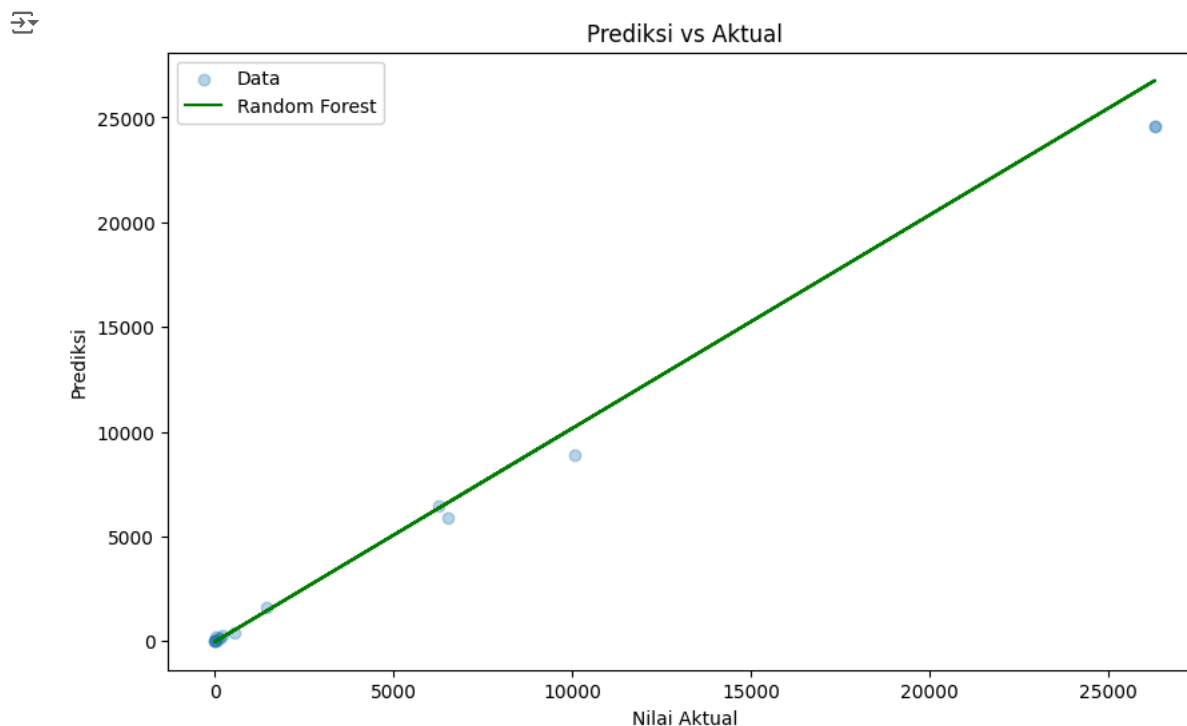
```

Skor untuk setiap fold: [0.995225211376509, 0.9958900523450581, 0.9898500226116429, 0.9663325813339106, 0.994922372285896, 0.988444]  
 Rata-rata skor R-squared: 0.8487065806322059  
 Standar deviasi skor R-squared: 0.3424178317458651

```

1 # Membuat plot
2 plt.figure(figsize=(10, 6))
3 plt.scatter(y_test, y_pred, alpha=0.3, label='Data')
4
5 # Menghitung garis regresi linear
6 coef = np.polyfit(y_test, lasso_predict, 1)
7 poly1d_fn = np.poly1d(coef)
8
9 # Menambahkan garis regresi linear pada plot
10 plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')
11
12 plt.xlabel('Nilai Aktual')
13 plt.ylabel('Prediksi')
14 plt.title('Prediksi vs Aktual')
15 plt.legend()
16 plt.show()

```






### ✓ 3.5 XGBoost Regression

```

1 import xgboost as xgb
2
3 # Inisialisasi model XGBoost
4 xgb_model = xgb.XGBRegressor(
5     n_estimators=1000,
6     learning_rate=0.01,
7     max_depth=30,
8     random_state=42
9 )
10
11 # Melatih model
12 xgb_model.fit(x_train, y_train)
13
14 # Memprediksi hasil pada data uji
15 xgb_predict = xgb_model.predict(x_test)

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


```

 R-Squared : 0.9942503854982147  
 RMSE : 654.0604731328712  
 MAE : 284.8176397362004

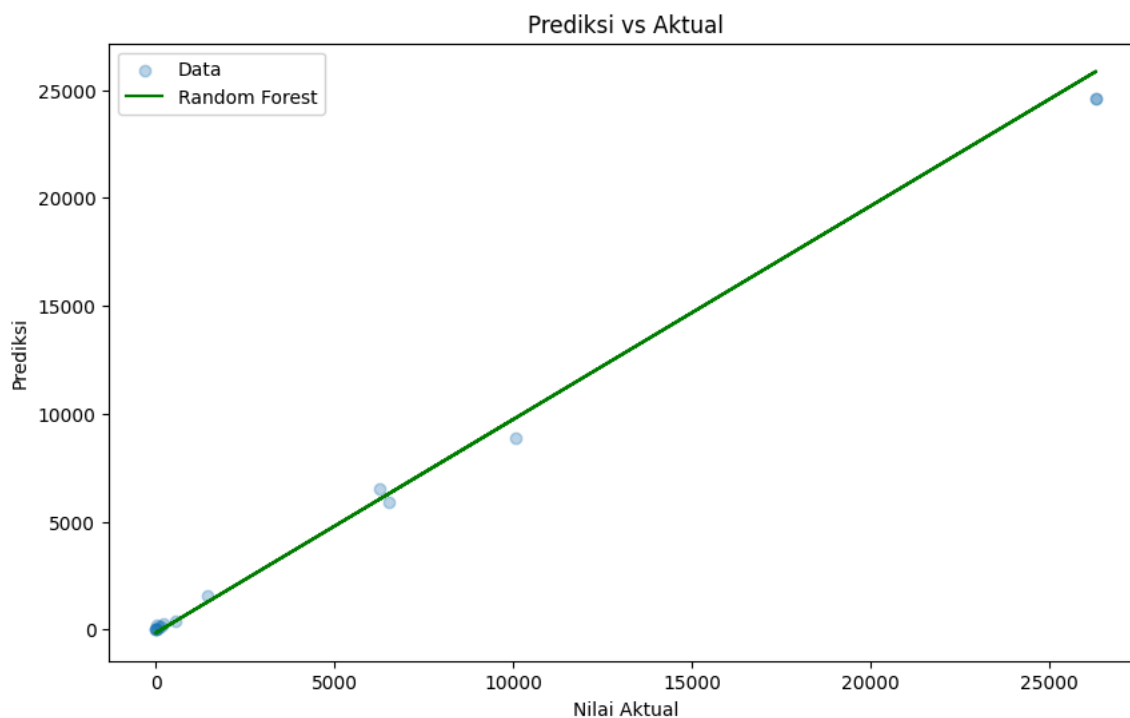
```

1 # K Fold xgb
2 kf_xgb = KFold(n_splits=5, shuffle=True, random_state=42)
3
4 xgb_r2_scores = []
5
6 xgb_model_kf = xgb.XGBRegressor(
7     n_estimators=1000,
8     learning_rate=0.01,
9     max_depth=30,
10    random_state=42
11 )
12
13 for train_index, test_index in kf_xgb.split(x):
14     x_train_fold, x_test_fold = x.iloc[train_index], x.iloc[test_index]
15     y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]
16
17     # Melatih model
18     xgb_model_kf.fit(x_train_fold, y_train_fold)
19
20     # Memprediksi hasil pada data uji
21     y_pred_fold = xgb_model_kf.predict(x_test_fold)
22
23     # Menghitung skor R-squared
24     r2 = r2_score(y_test_fold, y_pred_fold)
25     xgb_r2_scores.append(r2)
26
27 xgb_r2_scores.append(np.mean(xgb_r2_scores))
28 xgb_r2_scores.append(np.std(xgb_r2_scores))
29 # Menampilkan skor untuk setiap fold, rata-rata skor, dan standar deviasi
30 print("Skor untuk setiap fold: ", xgb_r2_scores)
31 print("Rata-rata skor R-squared: ", np.mean(xgb_r2_scores))
32 print("Standar deviasi skor R-squared: ", np.std(xgb_r2_scores))

```

 Skor untuk setiap fold: [0.9889089228708102, 0.9964807763991009, 0.982151804822498, 0.9571059965084702, 0.9739917656819577, 0.9797:  
 Rata-rata skor R-squared: 0.8415313487575345  
 Standar deviasi skor R-squared: 0.33870403875006105

```
1 # Membuat plot
2 plt.figure(figsize=(10, 6))
3 plt.scatter(y_test, y_pred, alpha=0.3, label='Data')
4
5 # Menghitung garis regresi linear
6 coef = np.polyfit(y_test, xgb_predict, 1)
7 poly1d_fn = np.poly1d(coef)
8
9 # Menambahkan garis regresi linear pada plot
10 plt.plot(y_test, poly1d_fn(y_test), color='green', label='Random Forest')
11
12 plt.xlabel('Nilai Aktual')
13 plt.ylabel('Prediksi')
14 plt.title('Prediksi vs Aktual')
15 plt.legend()
16 plt.show()
```

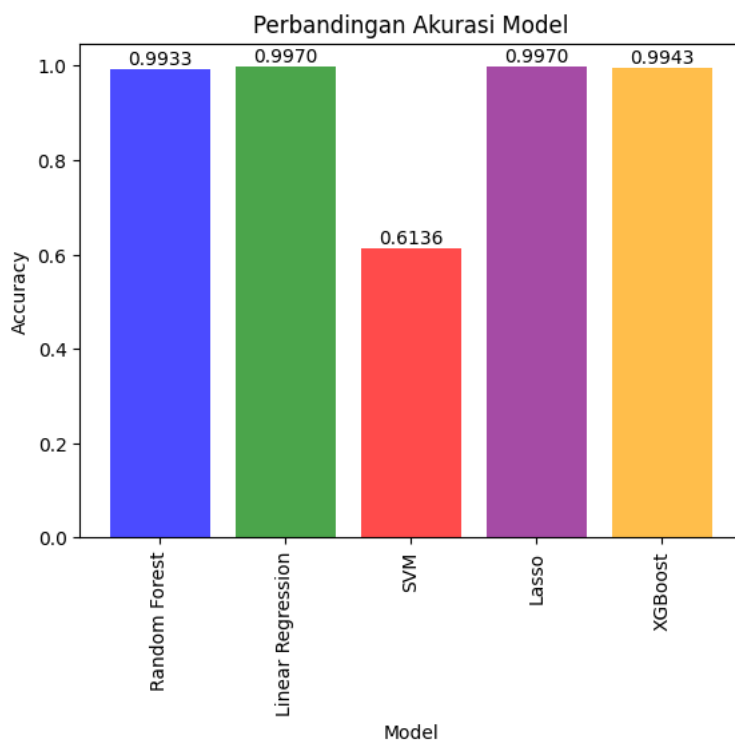


## ▼ Evaluasi

```

1 # Nama-nama model yang digunakan
2 model_names = ['Random Forest', 'Linear Regression', 'SVM', 'Lasso', 'XGBoost']
3
4 # Akurasi yang didapatkan dari masing-masing metode
5 accuracies = [
6     r2_score(y_test, y_pred),
7     r2_score(y_test, lr_predict),
8     r2_score(y_test, svr_predict),
9     r2_score(y_test, lasso_predict),
10    r2_score(y_test, xgb_predict),
11    # r2_score(y_test, ann_predict)
12 ]
13
14 # Menetapkan posisi batang di sumbu X
15 x_pos = np.arange(len(model_names))
16
17 # Membuat diagram batang
18 plt.bar(x_pos, accuracies, color=['blue', 'green', 'red', 'purple', 'orange'], alpha=0.7)
19
20 # Menambahkan nilai pada setiap batang
21 for i in range(len(accuracies)):
22     plt.text(x_pos[i], accuracies[i] + 0.01, f'{accuracies[i]:.4f}', ha='center')
23
24 # Menambahkan judul dan label
25 plt.xlabel('Model')
26 plt.ylabel('Accuracy')
27 plt.title('Perbandingan Akurasi Model')
28 plt.xticks(x_pos, model_names, rotation=90) # Menetapkan nama model sebagai label sumbu X
29
30 # Menampilkan plot
31 plt.show()
32
33 r_sq_df = pd.DataFrame({
34     'Model': model_names,
35     'R-Squared': accuracies
36 })
37 r_sq_df

```



	Model	R-Squared	
0	Random Forest	0.993299	
1	Linear Regression	0.996955	
2	SVM	0.613597	
3	Lasso	0.996955	
4	XGBoost	0.994250	

Next steps:

[Generate code with r\\_sq\\_df](#)
[View recommended plots](#)