```
!pip install openpyxl xgboost
```

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

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

1. Data Understanding

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

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 35995  100 35995    0     0  82905      0 --:--:-- --:--:-- --:--:-- 82747
```

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

df = pd.read_excel(file_path)
```

```
df
```

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

```
df.info()
```

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

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

```
1 df
```

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Station	Humidity
36	Cilacap	2013	1555	2093.392	13.462328	Kedelai	Meteorologi, Cilacap	82.0
37	Banyumas	2013	738	850.167	11.519878	Kedelai	NaN	NaN
38	Purbalingga	2013	129	204.291	15.836512	Kedelai	NaN	NaN
39	Banjarnegara	2013	277	319.155	11.521841	Kedelai	NaN	NaN
40	Kebumen	2013	3217	4539.329	14.110441	Kedelai	Sempor, Kebumen	84.0
...	...	...	...	...	...	...	...	..
505	Kota Surakarta	2018	0	0	0	Kedelai	NaN	NaN
506	Kota Salatiga	2018	5	7	15.53	Kedelai	NaN	NaN
507	Kota Semarang	2018	0	0	0	Kedelai	NaN	NaN

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

```
<ipython-input-125-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-125-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
66	Kota Magelang	2013	0	0	0	Kedelai	NaN	NaN
67	Kota Surakarta	2013	0	0	0	Kedelai	NaN	NaN
68	Kota Salatiga	2013	0	0	0	Kedelai	NaN	NaN
69	Kota Semarang	2013	0	0	0	Kedelai	NaN	NaN
70	Kota Pekalongan	2013	0	0	0	Kedelai	NaN	NaN
71	Kota Tegal	2013	0	0	0	Kedelai	NaN	NaN
246	Kota Magelang	2014	0	0	0	Kedelai	NaN	NaN
247	Kota Surakarta	2014	0	0	0	Kedelai	NaN	NaN
248	Kota Salatiga	2014	0	0	0	Kedelai	NaN	NaN
249	Kota Semarang	2014	0	0	0	Kedelai	NaN	NaN
250	Kota Pekalongan	2014	0	0	0	Kedelai	NaN	NaN
251	Kota Tegal	2014	0	0	0	Kedelai	NaN	NaN

```
1 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-129-95b6067be5a9>: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         76
dtype: int64
```

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

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         76
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-133-1c2a5001d4df>: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-133-1c2a5001d4df>: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-133-1c2a5001d4df>: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
36	Cilacap	2013	1555.0	2093.392	13.462328	Kedelai	82.0
37	Banyumas	2013	738.0	850.167	11.519878	Kedelai	NaN
38	Purbalingga	2013	129.0	204.291	15.836512	Kedelai	NaN
39	Banjarnegara	2013	277.0	319.155	11.521841	Kedelai	NaN
40	Kebumen	2013	3217.0	4539.329	14.110441	Kedelai	84.0

2.2 Interpolate Data NaN

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

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         76
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)
```

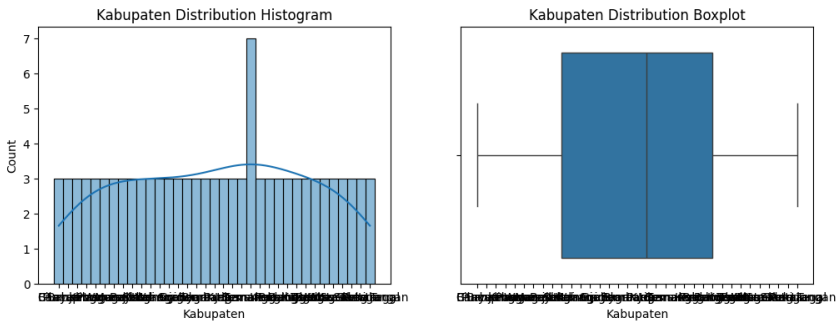
```
<ipython-input-136-ae5dbdddba06>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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-136-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-136-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-136-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)
```

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

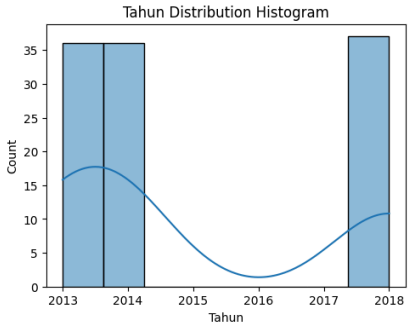
```
Kabupaten        0
Tahun            0
Luas Panen       0
Hasil Panen      0
Produktivitas    0
Tanaman          0
Humidity         0
dtype: int64
```

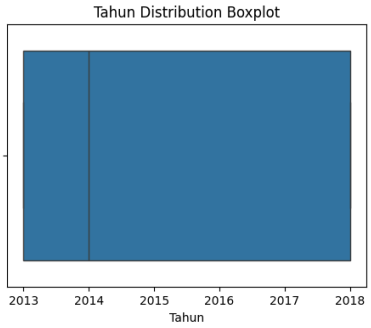
2.3 Handling outlier

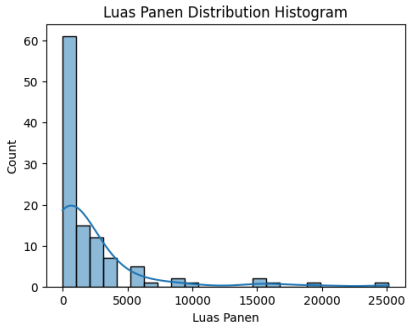
```
# Fungsi untuk menampilkan boxplot dan histogram
def num_dist(data, var):
    fig, ax = plt.subplots(1, 2, figsize=(12, 4))

    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)
```

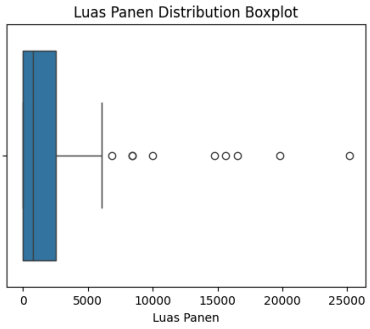

Kabupaten Distribution Histogram
Count
Kabupaten

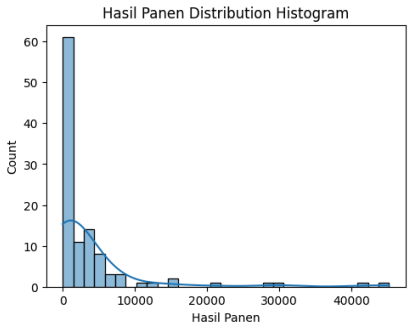
Kabupaten Distribution Boxplot
Kabupaten

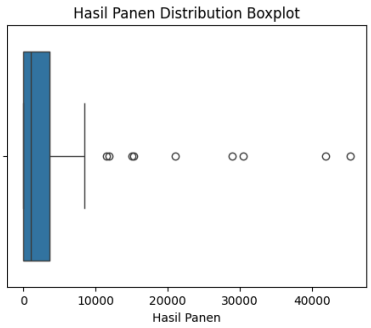

Tahun Distribution Histogram
Count
Tahun

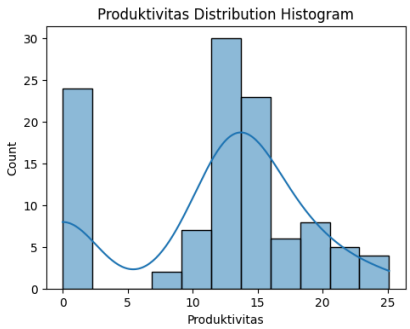

Tahun Distribution Boxplot
Tahun

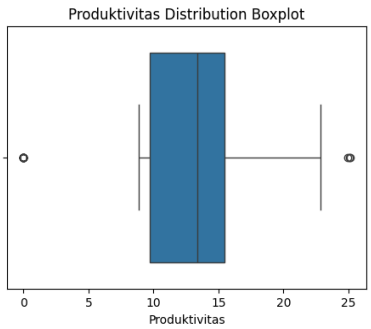

Luas Panen Distribution Histogram
Count
Luas Panen

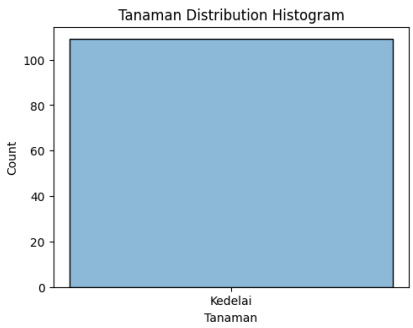

Luas Panen Distribution Boxplot
Luas Panen

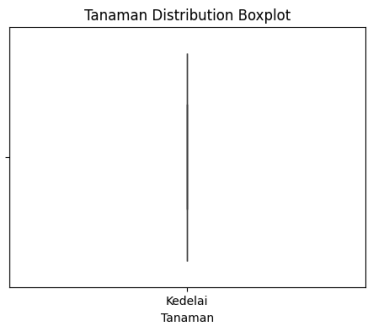

Hasil Panen Distribution Histogram
Count
Hasil Panen

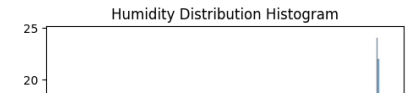

Hasil Panen Distribution Boxplot
Hasil Panen

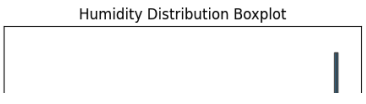

Produktivitas Distribution Histogram
Count
Produktivitas

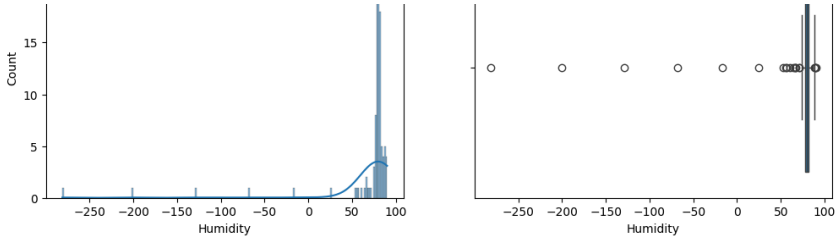

Produktivitas Distribution Boxplot
Produktivitas


Tanaman Distribution Histogram
Count
Kedelai
Tanaman


Tanaman Distribution Boxplot
Kedelai
Tanaman

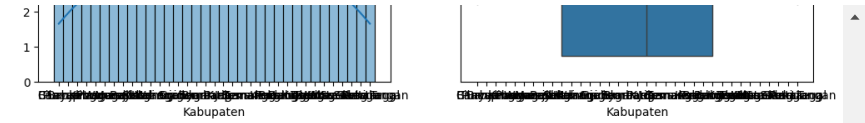

Humidity Distribution Histogram

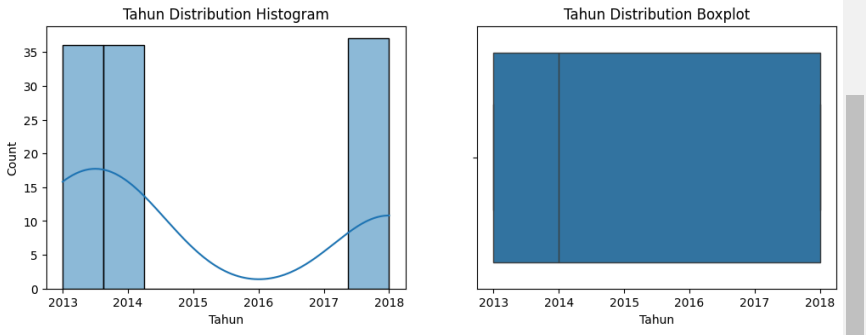

Humidity Distribution Boxplot

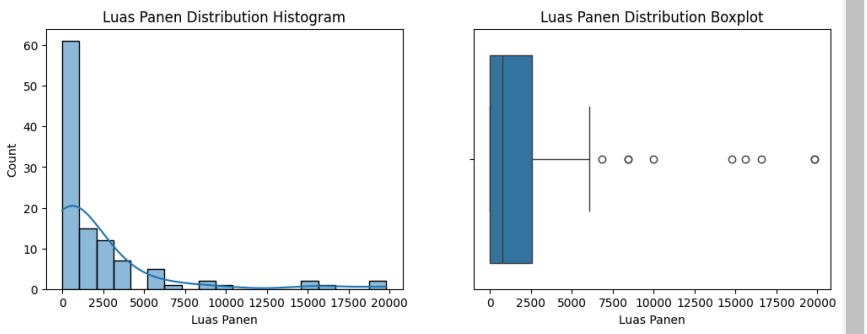


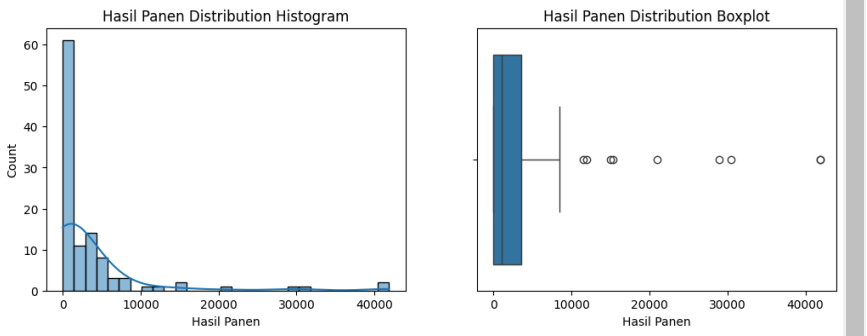

```
# Variabel yang terdapat outlier
outlier_var = ['Humidity', 'Produktivitas', 'Hasil Panen', 'Luas Panen']
threshold = 0.01

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

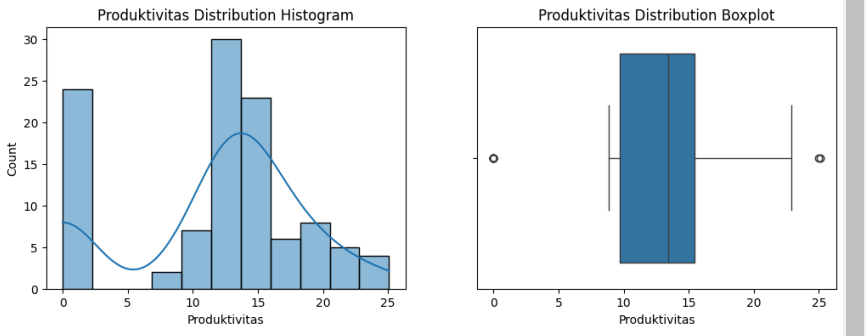
```
df_var = df.columns
for var in df_var:
    num_dist(df, var)
```

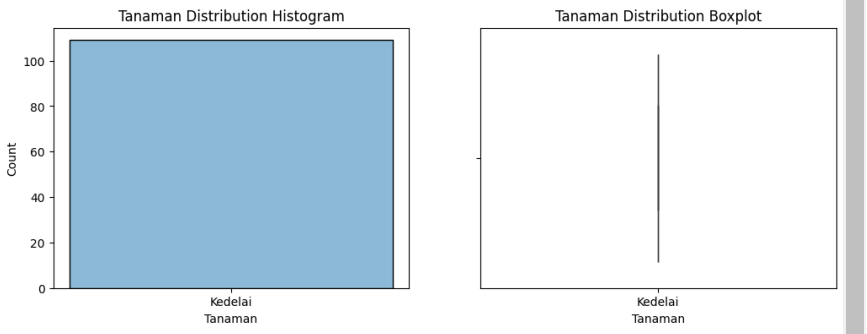

Kabupaten
Kabupaten

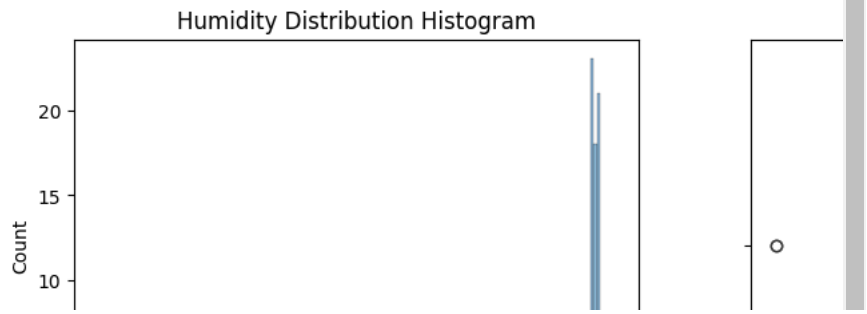

Tahun Distribution Histogram
Count
Tahun
Tahun Distribution Boxplot
Tahun

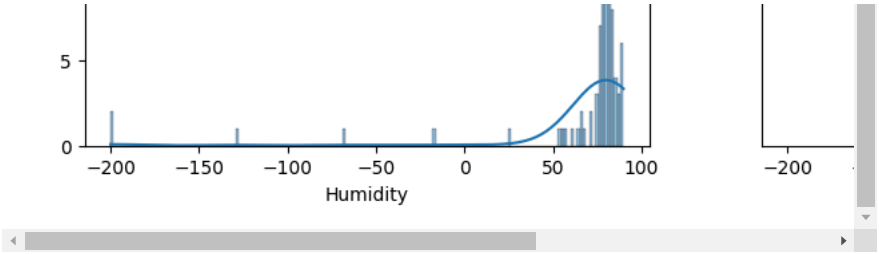

Luas Panen Distribution Histogram
Count
Luas Panen
Luas Panen Distribution Boxplot
Luas Panen


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
Kedelai
Tanaman
Tanaman Distribution Boxplot
Kedelai
Tanaman


Humidity Distribution Histogram
Count




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


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

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

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

	count	mean	std	min	25%	50%	75%	max
Luas Panen	109.0	2193.798165	3918.155870	0.000000	5.000000	738.000000	2556.00	19804.000000
Hasil Panen	109.0	3582.586532	7348.179820	0.000000	7.000000	1026.115000	3564.00	41866.000000
Produktivitas	109.0	11.633054	6.999664	0.000000	9.723143	13.400000	15.46	25.080769
Humidity	109.0	69.444562	45.813427	-199.984069	77.778577	79.931859	82.00	89.741623

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-141-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-141-d457133ed473>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

2.5 Visualisasi Data

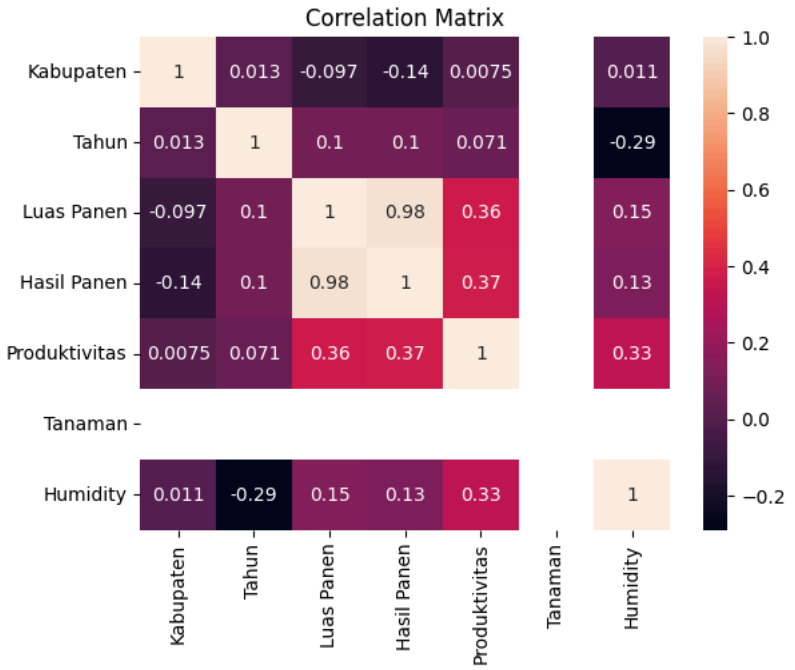
2.5 Correlation

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

	Kabupaten	Tahun	Luas Panen	Hasil Panen	Produktivitas	Tanaman	Humidi
Kabupaten	1.000000	0.013277	-0.097347	-0.137651	0.007530	NaN	0.0107
Tahun	0.013277	1.000000	0.102350	0.100836	0.070910	NaN	-0.2896
Luas Panen	-0.097347	0.102350	1.000000	0.978483	0.360517	NaN	0.1484
Hasil Panen	-0.137651	0.100836	0.978483	1.000000	0.370342	NaN	0.1293
Produktivitas	0.007530	0.070910	0.360517	0.370342	1.000000	NaN	0.3294
Tanaman	NaN	NaN	NaN	NaN	NaN	NaN	N

```
1 sns.heatmap(df.corr(), annot =True)
2 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 : (87, 3)
x_test : (22, 3)
y_train : (87,)
y_test : (22,)
```

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)
```

```
0.772186199608531
```

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

```
Luas Panen       0.962049
Produktivitas    0.019708
Humidity         0.018243
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	0.962049
1	Produktivitas	0.019708
2	Humidity	0.018243

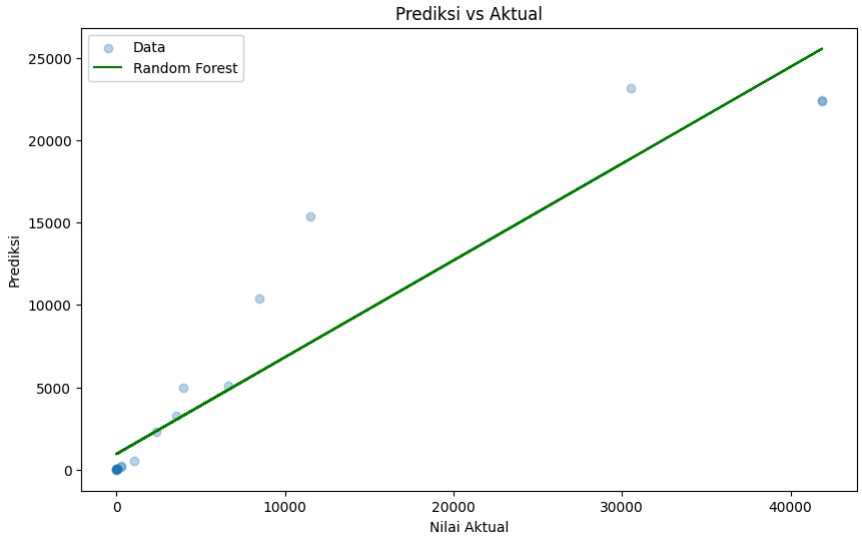
```
# K Fold RF
kf_rf = KFold(n_splits=5, shuffle=True, random_state=42)

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:  [0.9679688576610149, 0.943542488757226, 0.9662079627785258, 0.9723468005565238, 0.9047549869367831, 0.9509€
Rata-rata skor R-squared:  0.818395375159535
Standar deviasi skor R-squared:  0.3254223748186839
```

```
# 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 :  0.772186199608531
RMSE :  6158.217690833401
MAE :  2529.5755991735537
```

```
# 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
```

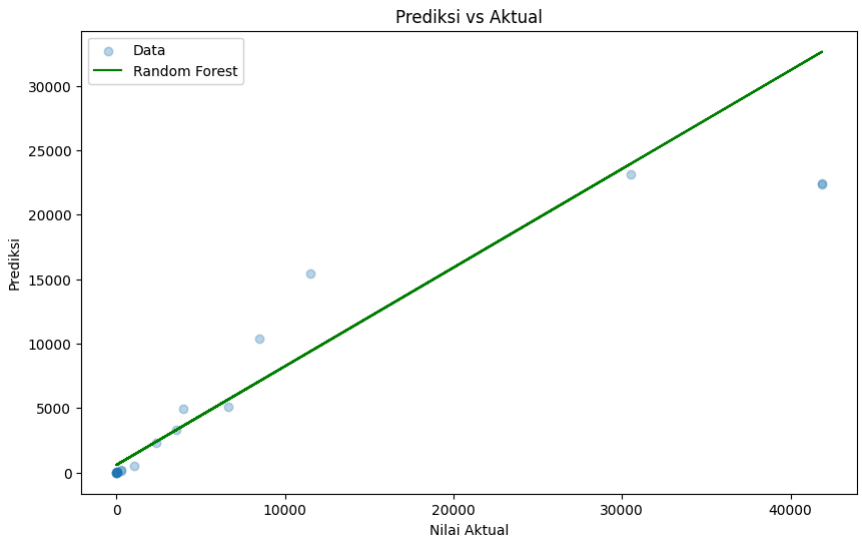
```
R-Squared :  0.9269615350009367
RMSE :  3486.910828061588
MAE :  1661.5464608677767
```

	Feature	Importance
0	Luas Panen	1.551685
1	Produktivitas	46.469316
2	Humidity	-2.952495

```
# 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:  [0.9557520617285239, 0.8037128867387998, 0.854853657245812, 0.9635325594158408, 0.9485473783304059, 0.90527
Rata-rata skor R-squared:  0.7843392250739845
Standar deviasi skor R-squared:  0.30118546126104623
```

```
# 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)
```

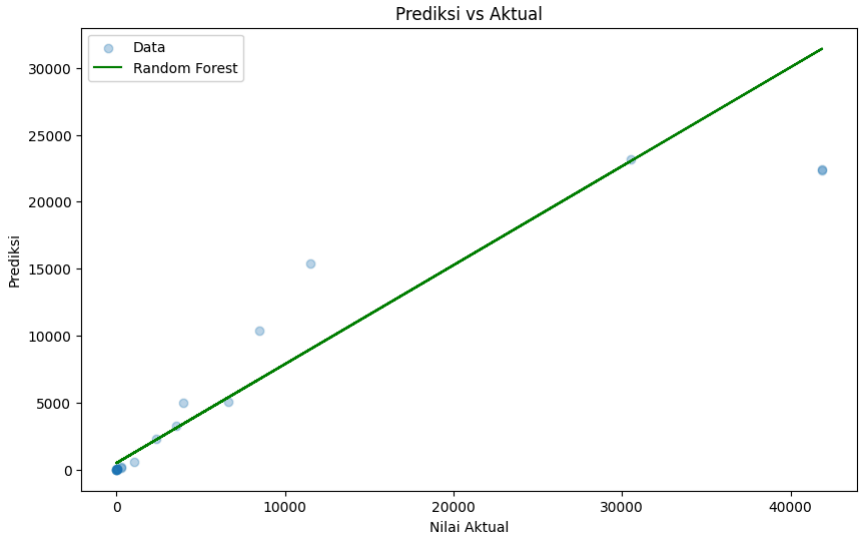
```
# K Fold SVR
kf_svr = KFold(n_splits=5, shuffle=True, random_state=42)

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:  [0.9219295972145788, 0.704106166334758, 0.9092080630826628, 0.9562834649094776, 0.9071713982519325, 0.87973
Rata-rata skor R-squared:  0.7657424534991629
Standar deviasi skor R-squared:  0.2893127206494363
```

```
# 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 :  0.9100701391052515
RMSE :  3869.162619173056
MAE :  1682.8040471574316
```

```
# 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)
```

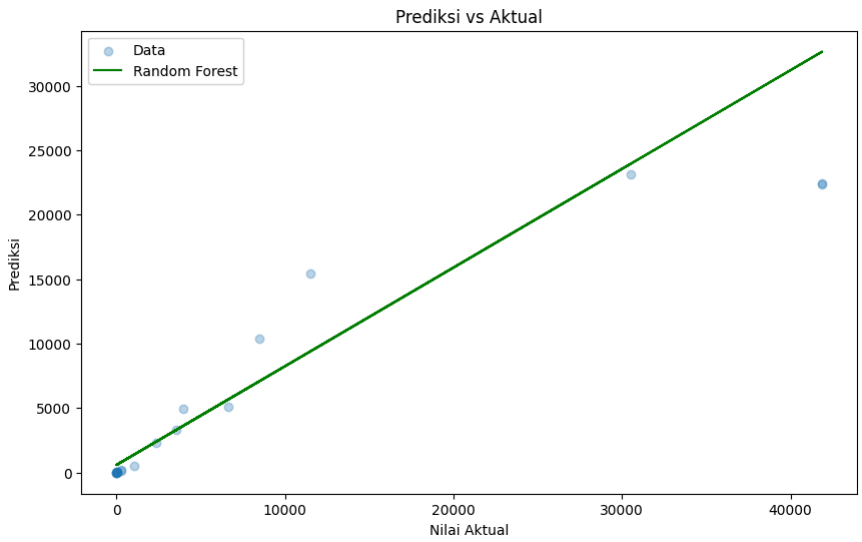
```
# Evaluation

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 :  0.9269615353047204
RMSE :  3486.9108208101575
MAE :  1661.5464487174927
```

```
# 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:  [0.9557520617291423, 0.8037128865164457, 0.8548536603806547, 0.9635325594392381, 0.9485473781944133, 0.9052
Rata-rata skor R-squared:  0.7843392254972894
Standar deviasi skor R-squared:  0.3011854615242345
```

```
# 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)
```

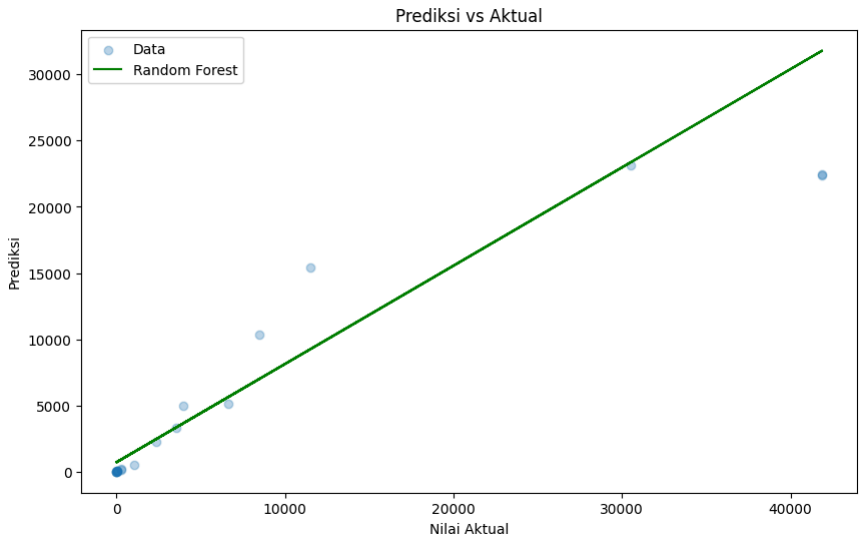
```
# Evaluation
print("R-Squared : ", r2_score(y_test, xgb_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, xgb_predict)))
print("MAE : ", mean_absolute_error(y_test, xgb_predict))
```

```
R-Squared :  0.9004460276192556
RMSE :  4070.9364180814578
MAE :  1650.0104896621706
```

```
# 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))
```

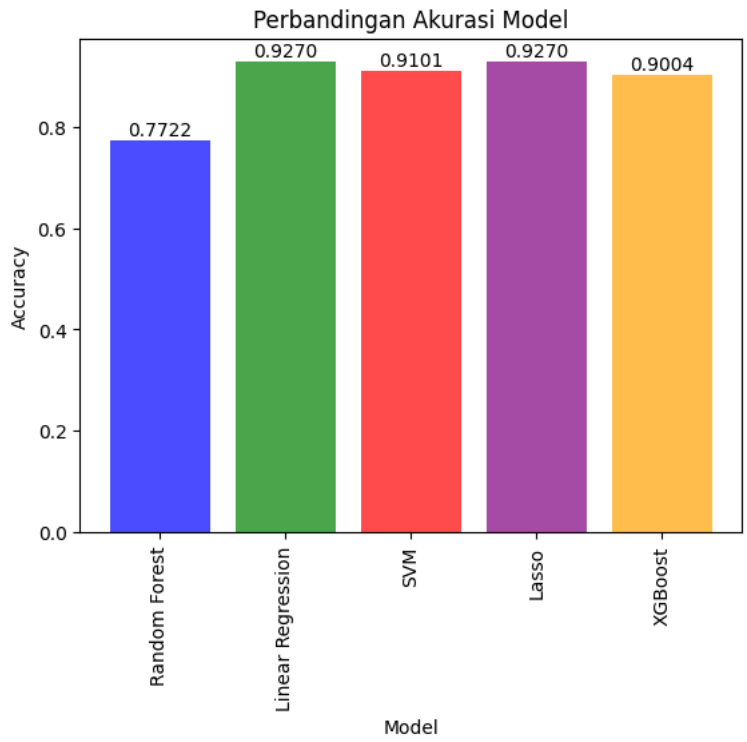
```
Skor untuk setiap fold:  [0.9604168141647006, 0.9418896008325626, 0.973217628797867, 0.9965692509751413, 0.9428908882174469, 0.9629
Rata-rata skor R-squared:  0.8280908207118627
Standar deviasi skor R-squared:  0.3309019241424429
```

```
# 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	0.772186
1	Linear Regression	0.926962
2	SVM	0.910070
3	Lasso	0.926962
4	XGBoost	0.900446