```
!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   125k      0 --:--:-- --:--:-- --:--:--  125k
```

```
# 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 == 'Kacang Tanah'")
```

```
1 df
```

| | Kabupaten | Tahun | Luas Panen | Hasil Panen | Produktivitas | Tanaman | Station | Humic |
|---|---|---|---|---|---|---|---|---|
| **0** | Cilacap | 2013 | 1322 | 1255.75075 | 9.498871 | Kacang Tanah | Meteorologi, Cilacap | |
| **1** | Banyumas | 2013 | 1671 | 2172.471503 | 13.001026 | Kacang Tanah | NaN | I |
| **2** | Purbalingga | 2013 | 731 | 780.888185 | 10.682465 | Kacang Tanah | NaN | I |
| **3** | Banjarnegara | 2013 | 2278 | 1970.754694 | 8.65125 | Kacang Tanah | NaN | I |
| **4** | Kebumen | 2013 | 2202 | 1938.738197 | 8.804442 | Kacang Tanah | Sempor, Kebumen | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **544** | Semarang | 2019 | 51 | 45 | 8,98 | Kacang Tanah | SMPK. Balit Getas, Semarang | |

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
  df['Kabupaten'] = df['Kabupaten'].str.replace('Kab. ', '')
<ipython-input-9-a905a613c6e2>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

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

| | Kabupaten | Tahun | Luas Panen | Hasil Panen | Produktivitas | Tanaman | Station | Humidity |
|---|---|---|---|---|---|---|---|---|
| **30** | Kota Magelang | 2013 | 0 | 0 | 0 | Kacang Tanah | NaN | NaN |
| **31** | Kota Surakarta | 2013 | 9 | 6.334032 | 7.037813 | Kacang Tanah | NaN | NaN |
| **32** | Kota Salatiga | 2013 | 0 | 0 | 0 | Kacang Tanah | NaN | NaN |
| **33** | Kota Semarang | 2013 | 341 | 422.075691 | 12.377586 | Kacang Tanah | NaN | NaN |
| **34** | Kota Pekalongan | 2013 | 0 | 0 | 0 | Kacang Tanah | NaN | NaN |
| **35** | Kota Tegal | 2013 | 0 | 0 | 0 | Kacang Tanah | NaN | NaN |
| **210** | Kota Magelang | 2014 | 0 | 0 | 0 | Kacang Tanah | NaN | NaN |

```
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 |
|---|---|---|---|---|---|---|---|---|

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

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

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

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

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         93
dtype: int64
```

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

```
Kabupaten         0
Tahun             0
Luas Panen        0
Hasil Panen       0
Produktivitas     0
Tanaman           0
Humidity         93
dtype: int64
```

# 2. Pre-Processing Data

## 2.1 Mengubah Tipe Data Variabel Menjadi Integer

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

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

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

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

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

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

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

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

| | Kabupaten | Tahun | Luas Panen | Hasil Panen | Produktivitas | Tanaman | Humidity |
|---|---|---|---|---|---|---|---|
| 0 | Cilacap | 2013 | 1322.0 | 1255.750750 | 9.498871 | Kacang Tanah | 82.0 |
| 1 | Banyumas | 2013 | 1671.0 | 2172.471503 | 13.001026 | Kacang Tanah | NaN |
| 2 | Purbalingga | 2013 | 731.0 | 780.888185 | 10.682465 | Kacang Tanah | 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        2
Hasil Panen       2
Produktivitas     2
Tanaman           0
Humidity         93
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)
```

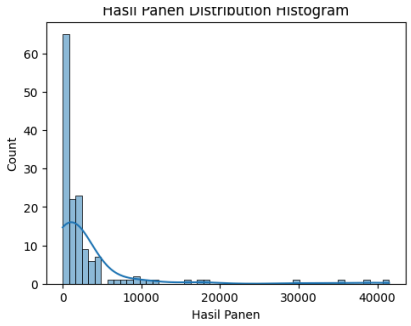
```
<ipython-input-20-ae5dbdddba06>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

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

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

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

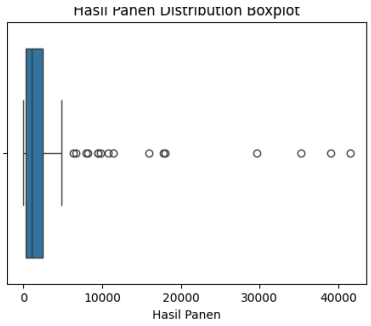
```
1 df.isna().sum()
```

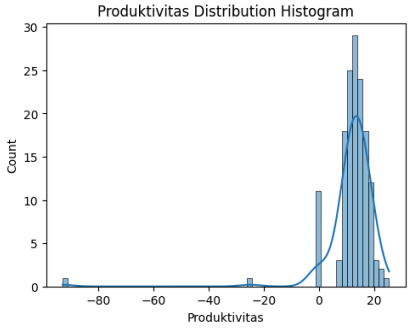
```
Kabupaten        0
Tahun            0
Luas Panen       0
Hasil Panen      0
Produktivitas    0
Tanaman          0
Humidity         0
dtype: int64
```

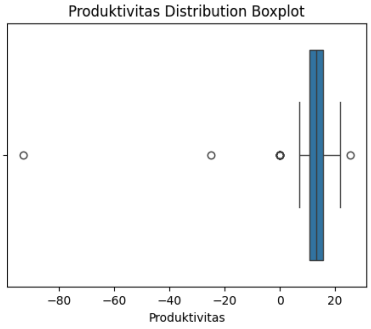
## 2.3 Handling outlier

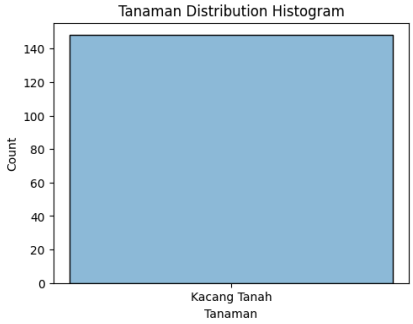
```
# Fungsi untuk menampilkan boxplot dan histogram
def num_dist(data, var):
    fig, ax = plt.subplots(1, 2, figsize=(12, 4))

    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)
```

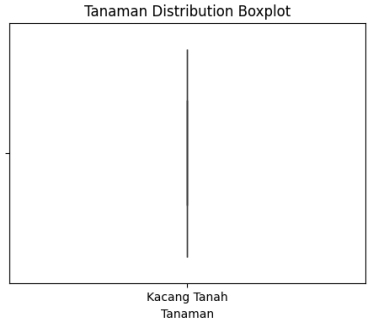

Hasil Panen Distribution Histogram
Count
Hasil Panen

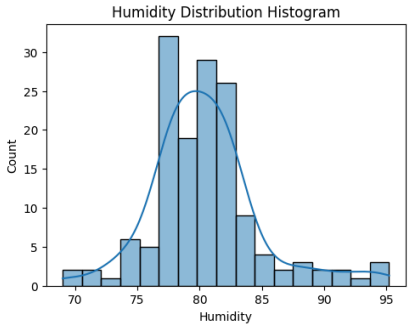

Hasil Panen Distribution Boxplot
Hasil Panen

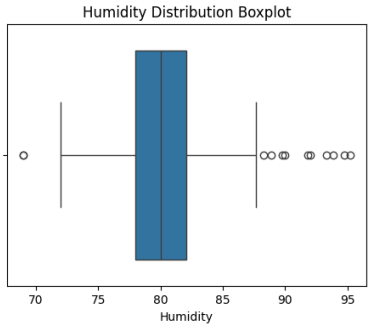

Produktivitas Distribution Histogram
Count
Produktivitas


Produktivitas Distribution Boxplot
Produktivitas


Tanaman Distribution Histogram
Count
Kacang Tanah
Tanaman

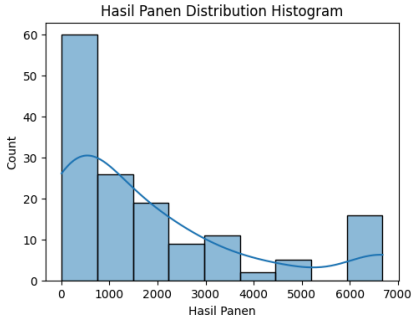

Tanaman Distribution Boxplot
Kacang Tanah
Tanaman

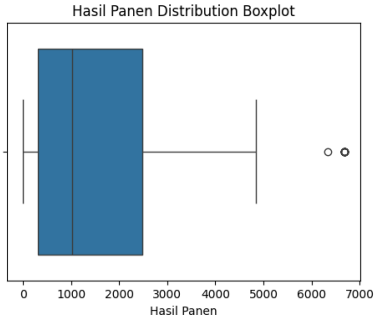

Humidity Distribution Histogram
Count
Humidity

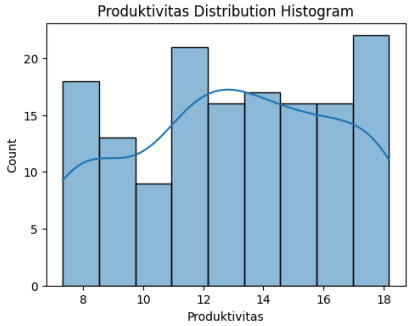

Humidity Distribution Boxplot
Humidity

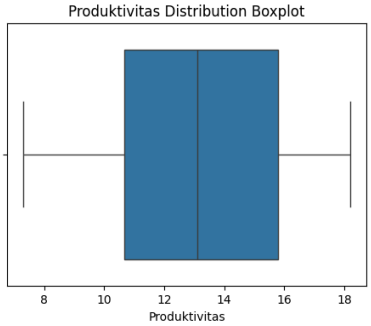
```
# Variabel yang terdapat outlier
outlier_var = ['Humidity', 'Produktivitas', 'Hasil Panen', 'Luas Panen']
threshold = 0.1

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

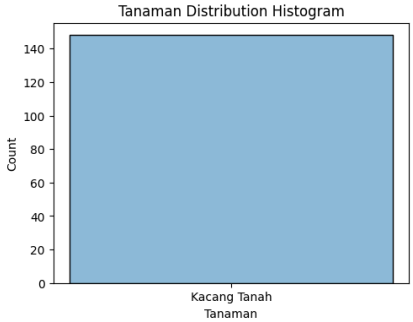
```
df_var = df.columns
for var in df_var:
    num_dist(df, var)
```

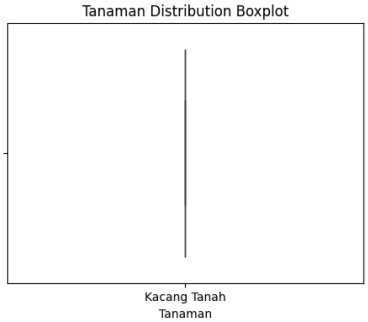

Hasil Panen Distribution Histogram
Count
Hasil Panen

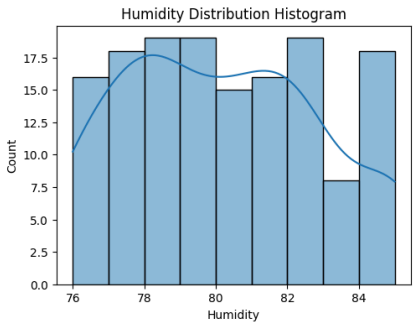

Hasil Panen Distribution Boxplot
Hasil Panen

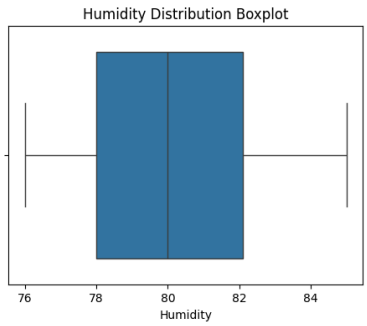

Produktivitas Distribution Histogram
Count
Produktivitas


Produktivitas Distribution Boxplot
Produktivitas


Tanaman Distribution Histogram
Count
Kacang Tanah
Tanaman


Tanaman Distribution Boxplot
Kacang Tanah
Tanaman


Humidity Distribution Histogram
Count
Humidity


Humidity Distribution Boxplot
Humidity

## 2.4 Encoding

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

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

## 2.5 Visualisasi Data

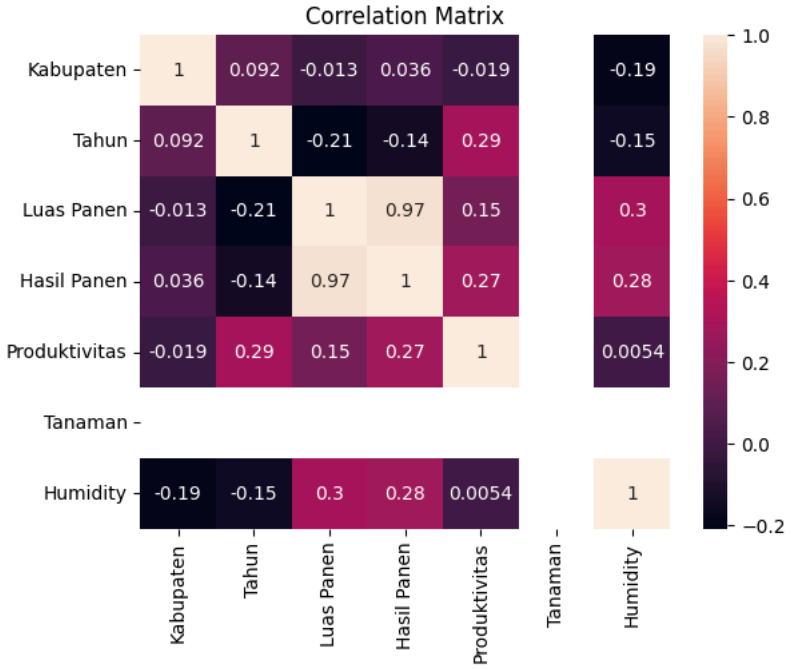
### 2.1 Correlation

```
df.corr()
```

| | Kabupaten | Tahun | Luas Panen | Hasil Panen | Produktivitas | Tanaman | Humidi |
|---|---|---|---|---|---|---|---|
| **Kabupaten** | 1.000000 | 0.092296 | -0.012516 | 0.036117 | -0.018503 | NaN | -0.1897 |
| **Tahun** | 0.092296 | 1.000000 | -0.207060 | -0.143035 | 0.293671 | NaN | -0.1519 |
| **Luas Panen** | -0.012516 | -0.207060 | 1.000000 | 0.973389 | 0.151716 | NaN | 0.2958 |
| **Hasil Panen** | 0.036117 | -0.143035 | 0.973389 | 1.000000 | 0.274405 | NaN | 0.2761 |
| **Produktivitas** | -0.018503 | 0.293671 | 0.151716 | 0.274405 | 1.000000 | NaN | 0.0054 |
| **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 : (118, 3)
x_test : (30, 3)
y_train : (118,)
```

```
y_test : (30,)
```

## 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)
```

```
0.9844814593155349
```

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

```
Luas Panen       0.979044
Produktivitas    0.016031
Humidity         0.004924
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 | 0.979044 |
| 1 | Produktivitas | 0.016031 |
| 2 | Humidity | 0.004924 |

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))
```

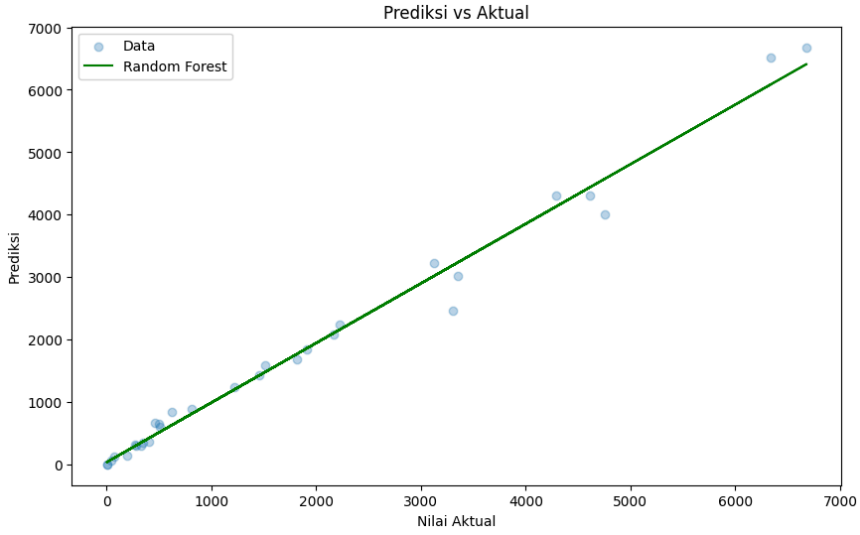
```
Skor untuk setiap fold:  [0.9932825804273104, 0.9735875482147619, 0.9642645203488016, 0.9933037015234356, 0.956657789519172, 0.97621
Rata-rata skor R-squared:  0.8387072378060615
```

```
Standar deviasi skor R-squared:  0.3370706837140175
```

```
# 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 :  0.9844814593155349
MAE :  131.13332909766552
```

```
# 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 :  0.9696699300148313
RMSE :  330.2817590256926
MAE :  252.0215132106151
```

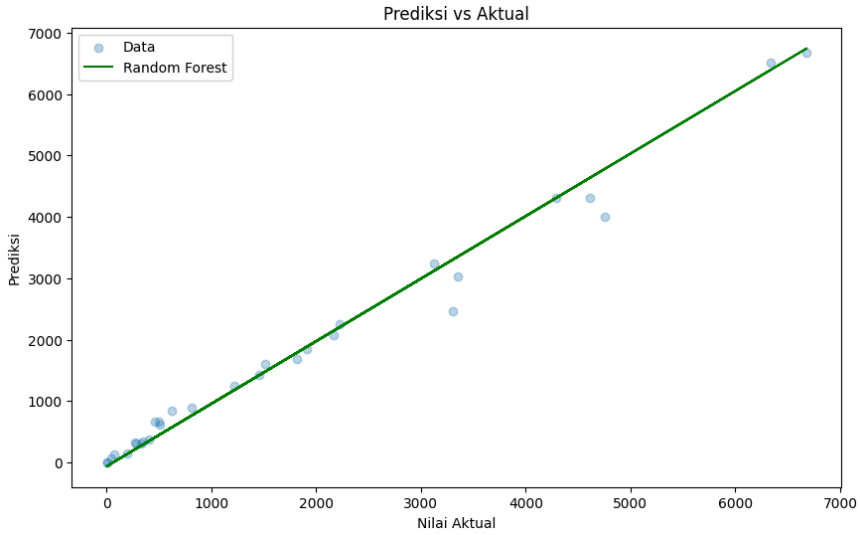
| | Feature | Importance |
|---|---|---|
| 0 | Luas Panen | 1.440532 |
| 1 | Produktivitas | 79.265311 |
| 2 | Humidity | -16.277573 |

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:  [0.9734570639008593, 0.9605138870813071, 0.9523517547374676, 0.9825296172377677, 0.880327246804688, 0.9498
Rata-rata skor R-squared:  0.8188754314204819
Standar deviasi skor R-squared:  0.32224787613240974
```

```
# 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)
```

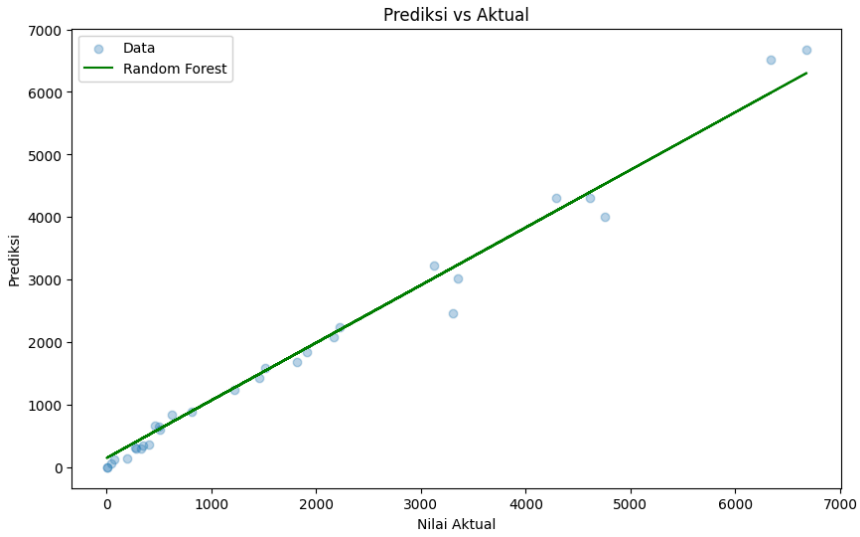
```
# K Fold SVR
kf_svr = KFold(n_splits=5, shuffle=True, random_state=42)

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:  [0.9704258112196256, 0.9427324433503375, 0.9394079611955506, 0.8329510659297069, 0.6703888880653068, 0.8711
Rata-rata skor R-squared:  0.7611785519622362
Standar deviasi skor R-squared:  0.28526374808385824
```

```
# 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 :  0.9655783904741532
RMSE :  351.8547915012077
MAE :  234.15011048090904
```

```
# 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)
```

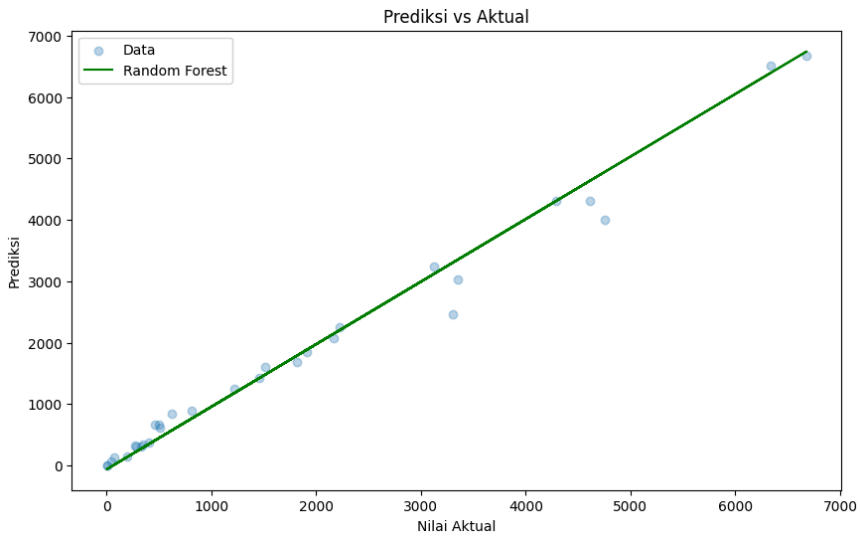
```
# Evaluation
print("R-Squared : ", r2_score(y_test, lasso_predict))
print("RMSE : ", np.sqrt(mean_squared_error(y_test, lasso_predict)))
print("MAE : ", mean_absolute_error(y_test, lasso_predict))
```

```
R-Squared :  0.9696699338389426
RMSE :  330.2817382042061
MAE :  252.02149686360318
```

```
# 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:  [0.9734570648751213, 0.9605138854612353, 0.9523517570584784, 0.9825296154145009, 0.8803272459236284, 0.949
Rata-rata skor R-squared:  0.8188754312535416
Standar deviasi skor R-squared:  0.3222478760434398
```

```
# 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))
```

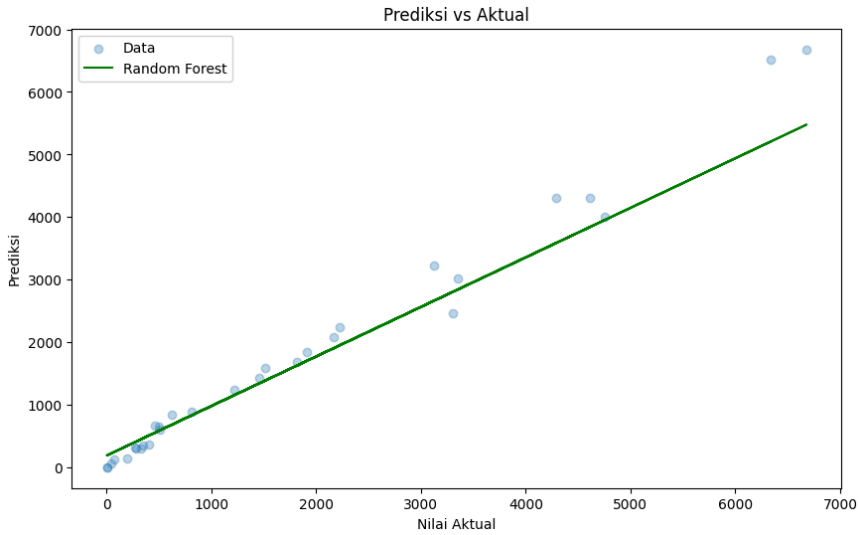
```
R-Squared :  0.9093182940942519
RMSE :  571.0941657094437
MAE :  238.77192827709885
```

```
# 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:  [0.9498472322859244, 0.979891587469369, 0.973890101485517, 0.99678009834308, 0.9643107996923659, 0.9729439
Rata-rata skor R-squared:  0.8359934741410392
Standar deviasi skor R-squared:  0.33571962261784893
```

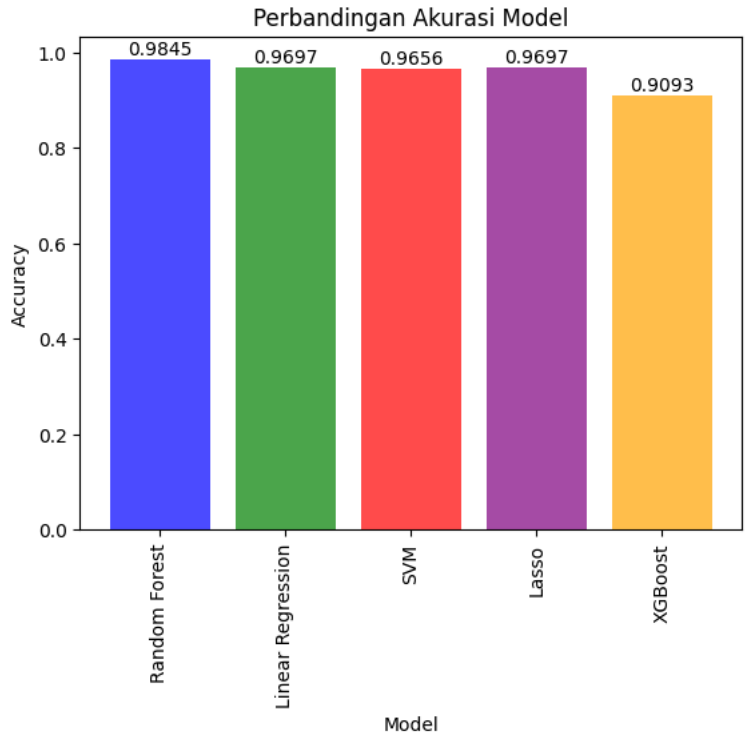
```
# Membuat plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3, label='Data')

# 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()
```

```
15 plt.legend()
16 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 | 0.984481 |
| 1 | Linear Regression | 0.969670 |
| 2 | SVM | 0.965578 |
| 3 | Lasso | 0.969670 |
| 4 | XGBoost | 0.909318 |

Next steps: Generate code with r_sq_df View recommended plots