Iris Flower : EDA and Prediction

Utus Karta Sanggam

Outline

1. Introduction
2. Goals
3. Data Understanding
4. Tools yang digunakan
5. Data Analysis
6. Data Modelling
7. Conclusion
8. My Profile

Introduction: Iris flower Dataset

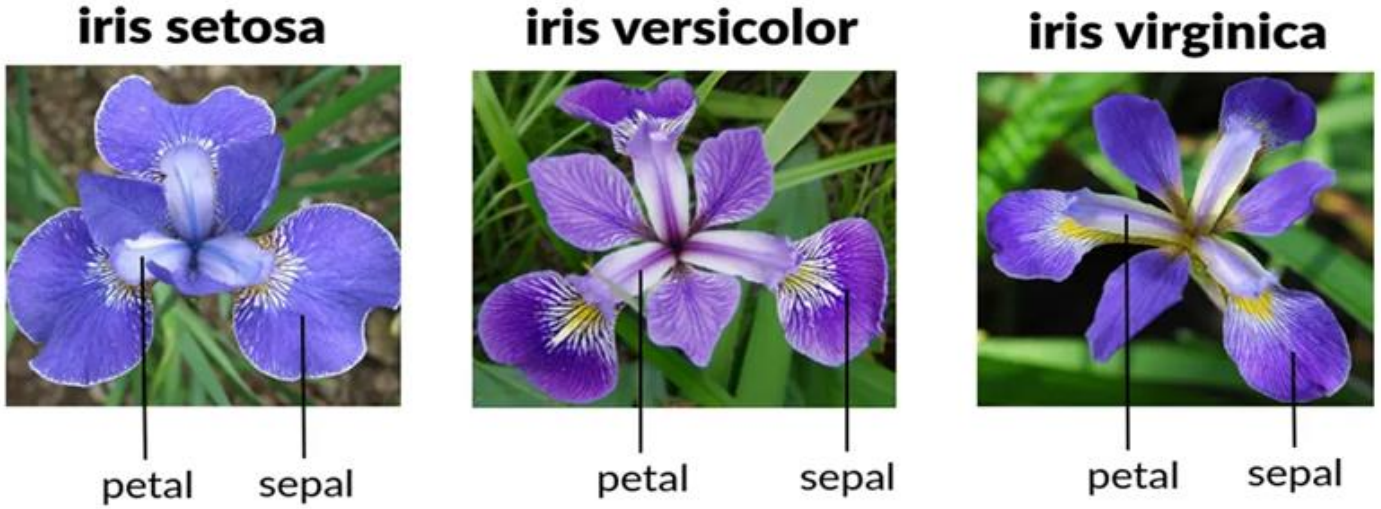
The ***Iris* flower data set** or **Fisher's *Iris* data set** is a multivariate data set used and made famous by the British statistician and Biologist Ronald Fisher in his 1936 paper *The use of multiple measurements in taxonomic problems* as an example of linear discriminant analysis.[1] It is sometimes called **Anderson's *Iris* data set** because collected the data to quantify the morphologic variation of *Iris* flowers of three related species.[2] Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus".[3]

The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other. Fisher's paper was published in the *Annals of Eugenics* (today the *Annals of Human Genetics*) and includes discussion of the contained techniques' applications to the field of phrenology.[1]

Problem Identification

Goal: Menentukan jenis spesies Bunga Irish

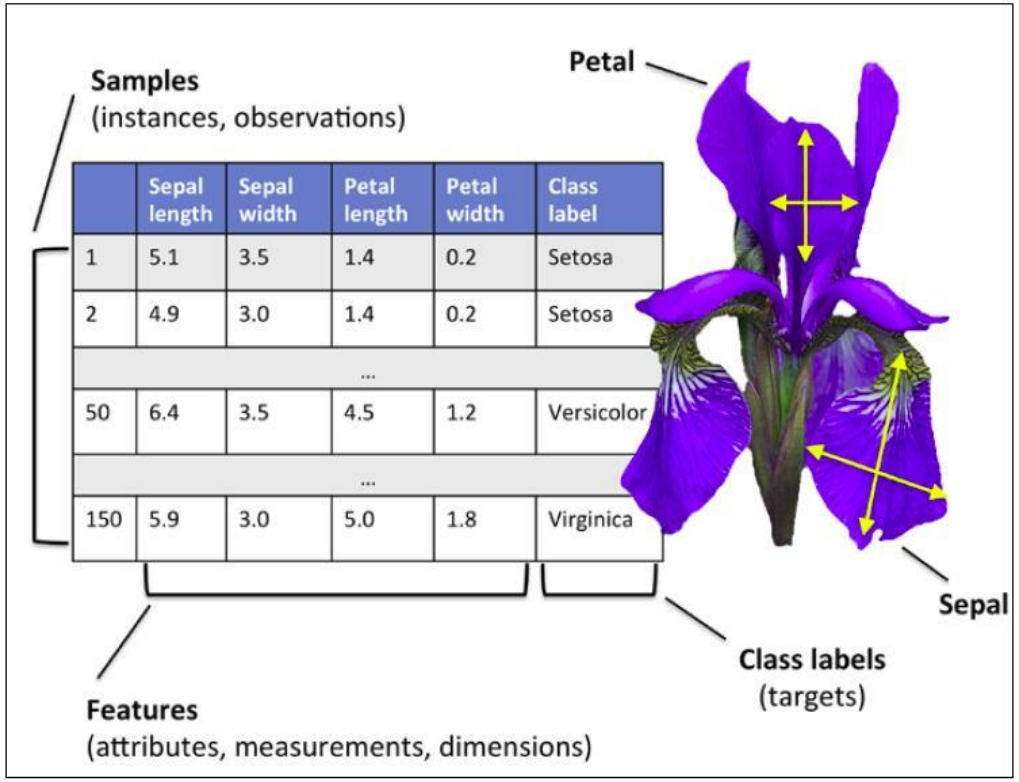
Ciri khas bunga iris yang dapat dibedakan berdasarkan panjang dan lebar sepal dan petal. Setiap kumpulan data terdiri dari penomoran berkelanjutan, empat karakteristik dan penugasan ke salah satu (kelas) umum. Kumpulan data pertama yang disebut “kumpulan data Iris” dapat dihasilkan sebagai array dengan enam entri. Klasifikasi pola harus dibuat yang membedakan dua atau ketiga spesies iris (setosa, versicolor, viriginica) berdasarkan sepal dan petal.



Iris Flower Dataset

Data Understanding: Dataset Bunga Iris

- Atribut : Sepal Length, Sepal Width, Petal Length, Petal Width.
- Jumlah data : 150
- Spesies : Iris-setosa (50), Iris-versicolor (50), Iris-virginica (50).
- Sepal merupakan bagian pembentuk kelopak bunga, biasanya berfungsi sebagai pelindung bunga saat kuncup, dan sering kali sebagai penopang kelopak bunga saat mekar.
- Petal merupakan kelopak daun bunga sebagai hasil modifikasi daun yang mengelilingi bagian reproduksi bunga.



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Tools yang digunakan

Google Colaboratory atau **Google Colab** adalah executable document yang memungkinkan untuk dalam menulis, mengedit, serta membagikan program yang sudah disimpan pada gogle drive.

Python merupakan bahasa pemrograman komputer yang biasa dipakai untuk membangun situs, software/aplikasi, mengotomatiskan tugas dan melakukan analisis data. Bahasa pemrograman ini termasuk bahasa tujuan umum.

Python Libraries merupakan pendukung untuk membangun situs, software/aplikasi, mengotomatiskan tugas dan melakukan data analysis, , visualisasi, membuat model dan prediksi seperti Pandas, Scikit-Learn, Matplotlib, NumPy, Seaborn, Streamlit.

Data Quality (1/4)

Melakukan pengecekan data, missing/null value, duplikat, outlier.

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
df.tail()
```

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

```
print('Rows:',df.shape[0])
print('Columns:',df.shape[1])
print()

print('Features')
print()
print( df.columns.tolist())
print()

print(' Unique Values')
print()
print( df.nunique())
```

```
Rows: 150
Columns: 5

Features

['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']

 Unique Values

sepal_length    35
sepal_width     23
petal_length    43
petal_width     22
species          3
dtype: int64
```

Jumlah Dataset ada :
150 Baris x 5 Kolom

Data Quality (2/4)

Melakukan pengecekan `data`, `missing/null value`, duplikat, outlier.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

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

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
dtype: int64
```

```
df.duplicated().sum()
```

```
1
```

```
df.value_counts()
```

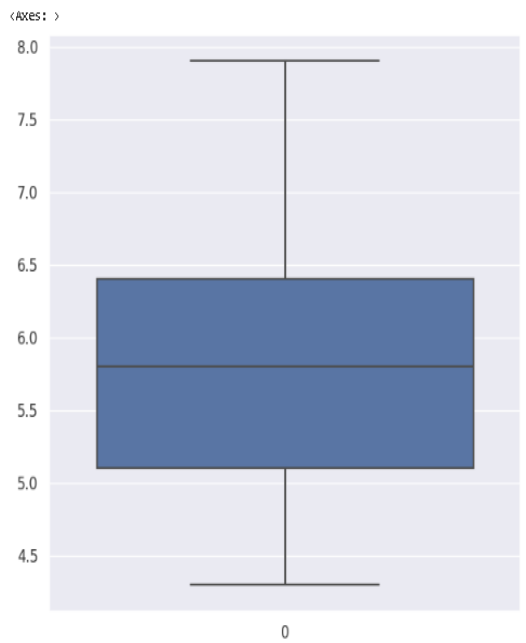
```
sepal_length  sepal_width  petal_length  petal_width
5.8           2.7          5.1           1.9          2
6.2           2.2          4.5           1.5          1
              2.9          4.3           1.3          1
              3.4          5.4           2.3          1
6.3           2.3          4.4           1.3          1
                                                      ..
5.4           3.9          1.3           0.4          1
                           1.7           0.4          1
5.5           2.3          4.0           1.3          1
              2.4          3.7           1.0          1
7.9           3.8          6.4           2.0          1
Length: 149, dtype: int64
```

Ada Duplikat Data pada Baris Pertama

Data Quality (3/4)

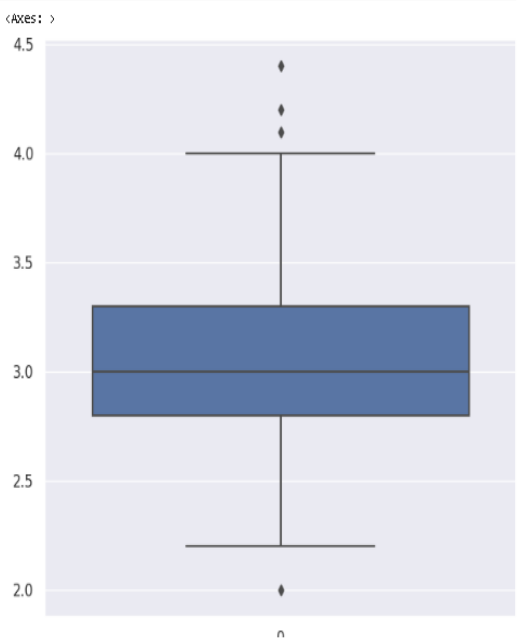
Melakukan pengecekan data, missing/null value, duplikat, outlier.

```
plt.figure(figsize=(7,7))
sns.boxplot(df['sepal_length'])
```



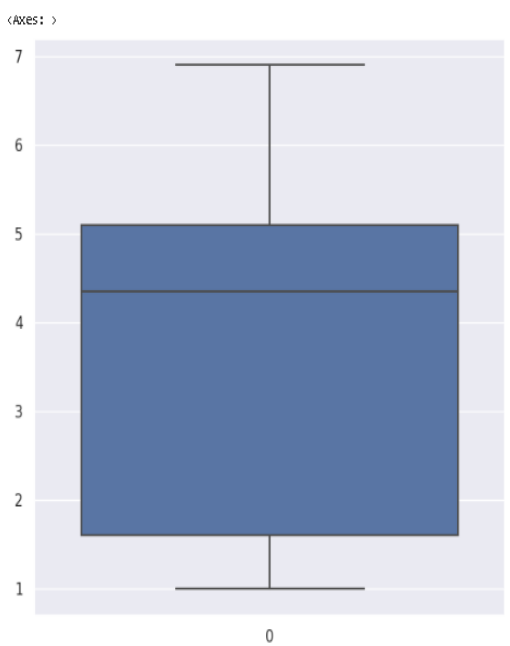
Sepal Length

```
plt.figure(figsize=(7,7))
sns.boxplot(df['sepal_width'])
```



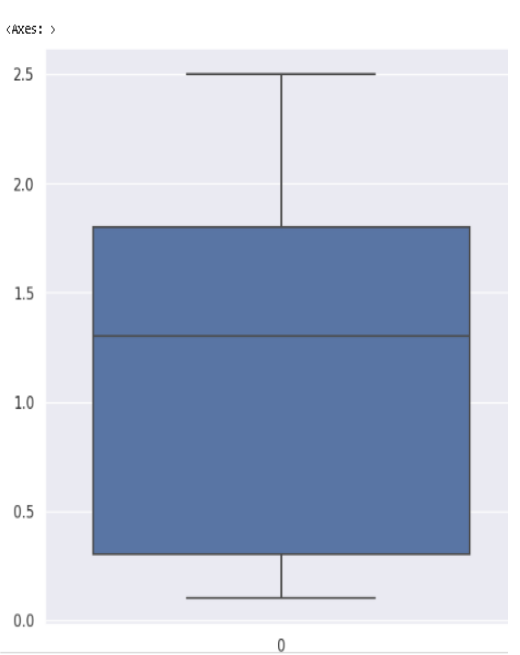
Sepal Width

```
plt.figure(figsize=(7,7))
sns.boxplot(df['petal_length'])
```



Petal Length

```
plt.figure(figsize=(7,7))
sns.boxplot(df['petal_width'])
```



Petal Width

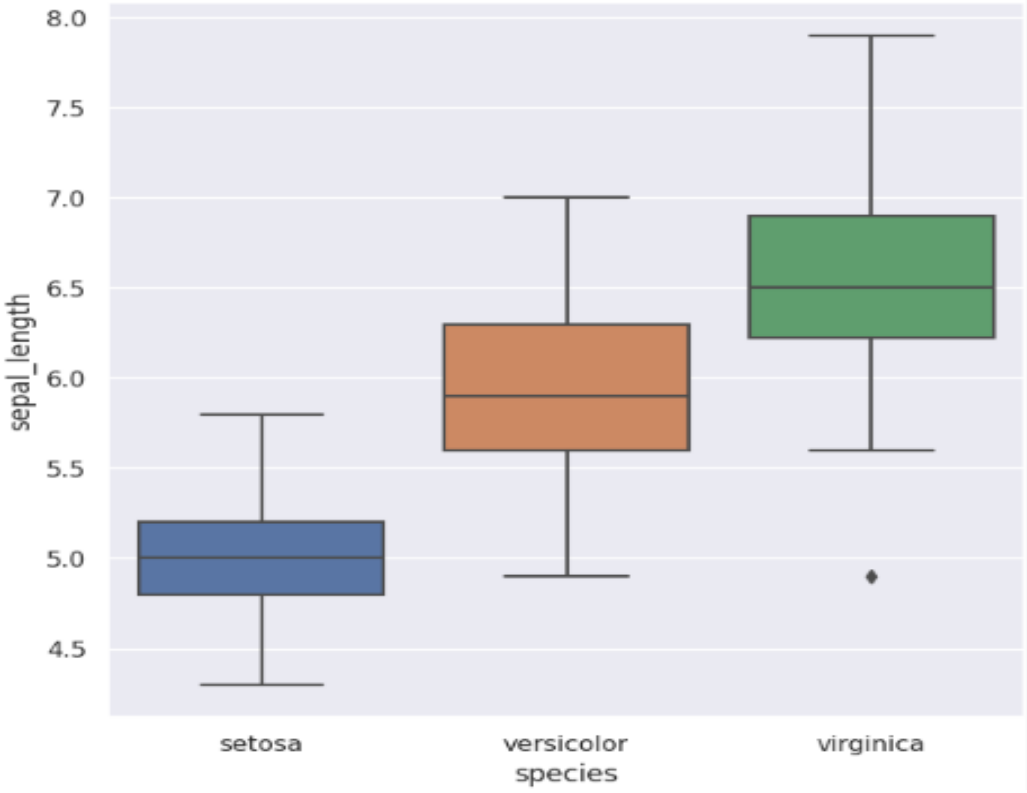
Data yang memiliki outlier dengan menggantikan dengan

1. Nilai Max: Q3 + 1.5 IQR
2. Nilai Min: Ql-1.5 IQR

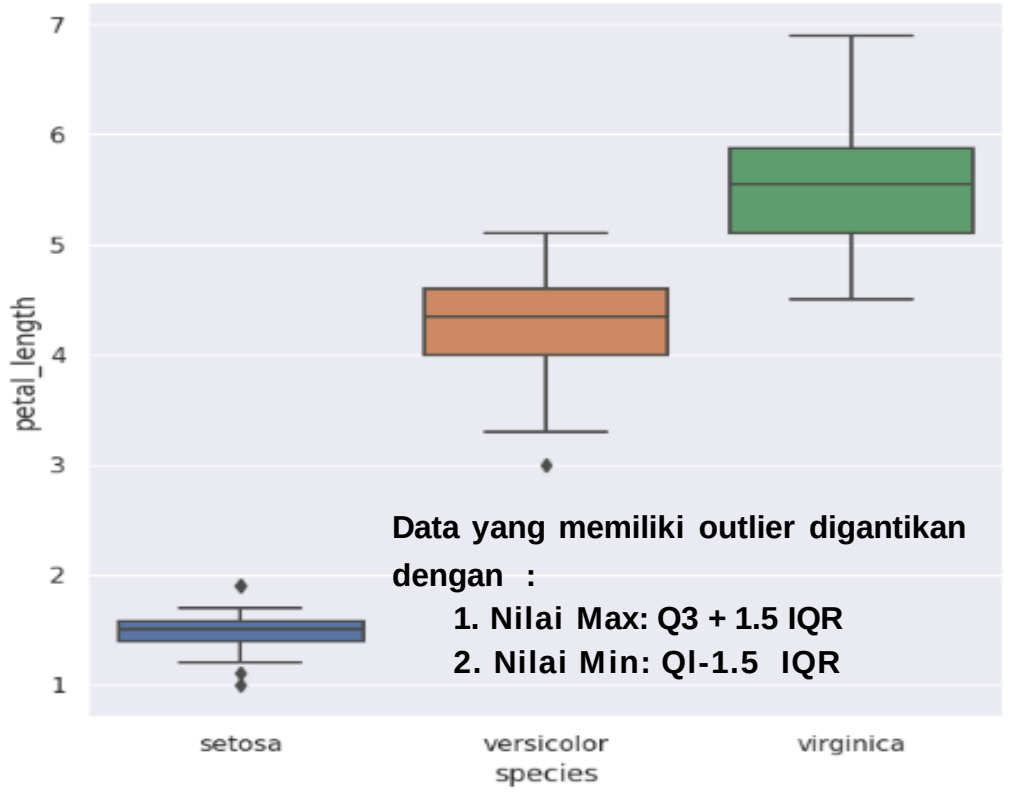
Data Quality (4/4)

Melakukan pengecekan data, missing/null value, duplikat, outlier.

```
plt.figure(figsize=(7,7))
sns.boxplot(x='species', y='sepal_length', data=df)
sns.set(rc = {'figure.figsize':(25,10)})
plt.show()
```




```
plt.figure(figsize=(7,7))
sns.boxplot(x='species', y='petal_length', data=df)
sns.set(rc = {'figure.figsize':(25,10)})
plt.show()
```

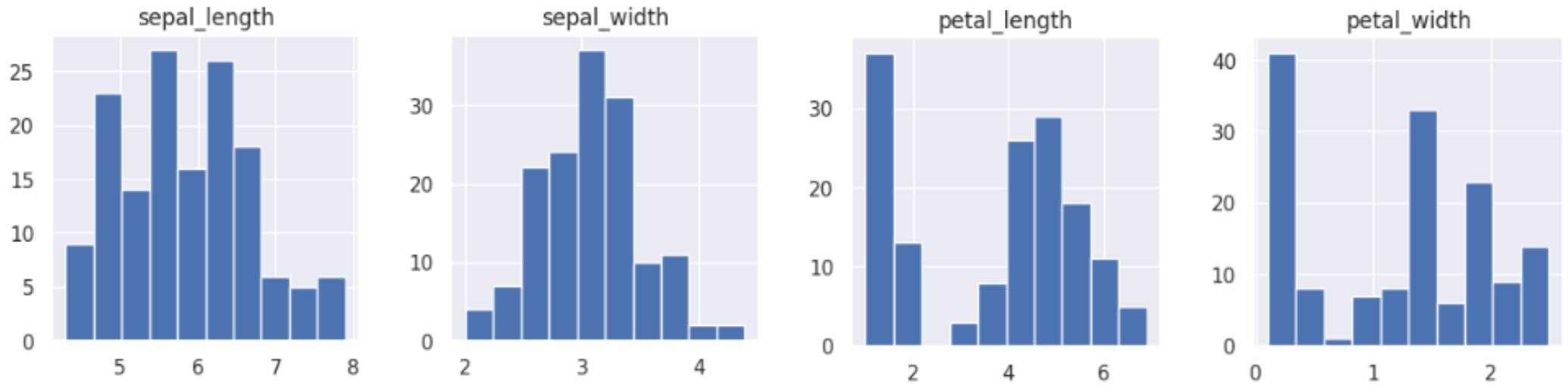


Data yang memiliki outlier digantikan dengan :

1. **Nilai Max: Q3 + 1.5 IQR**
2. **Nilai Min: Ql-1.5 IQR**

Data Analysis (1/6)

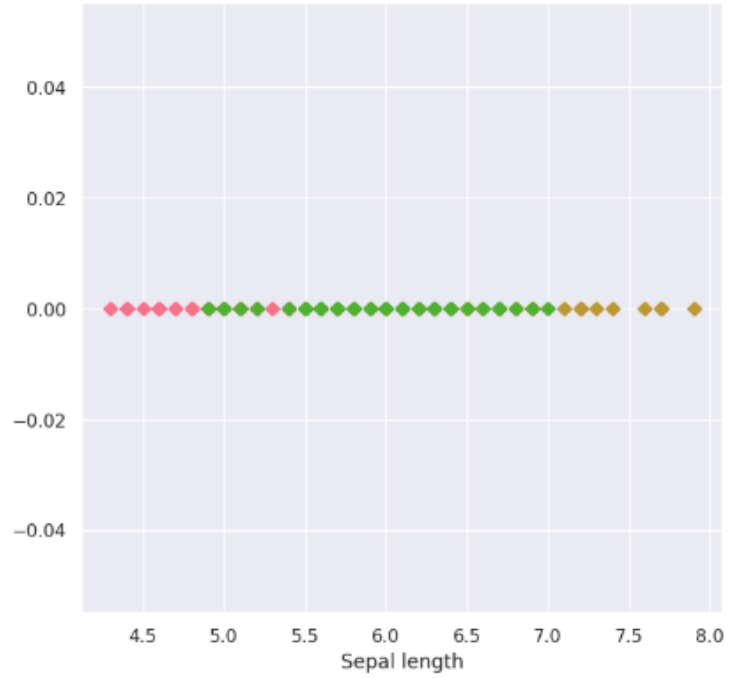
```
df.hist(figsize=(7,7))
plt.show()
```



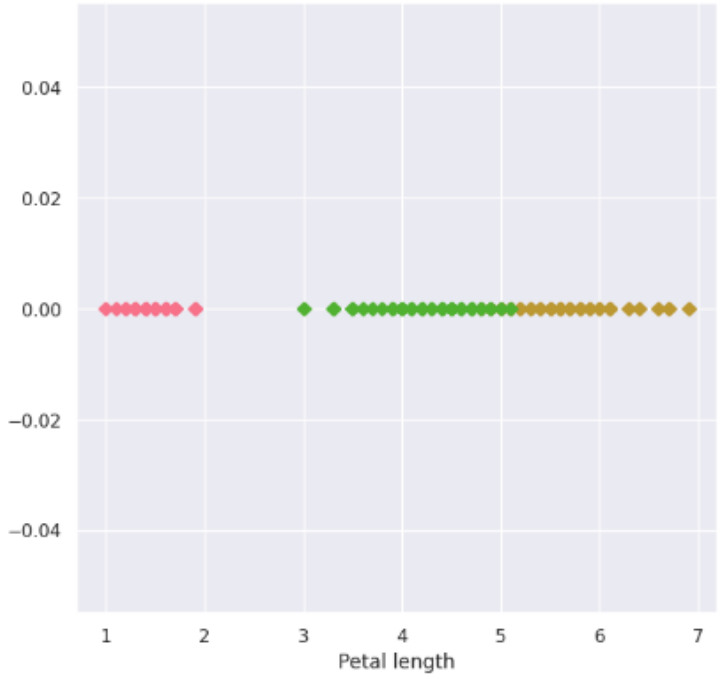
Data Analysis (2/6)

Uni Variate Analysis

```
plt.figure(figsize=(7,7))
plt.plot(df_setosa['sepal_length'], np.zeros_like(df_setosa['sepal_length']), 'D')
plt.plot(df_virginica['sepal_length'], np.zeros_like(df_virginica['sepal_length']), 'D')
plt.plot(df_versicolor['sepal_length'], np.zeros_like(df_versicolor['sepal_length']), 'D')
plt.xlabel('Sepal length')
plt.show()
```



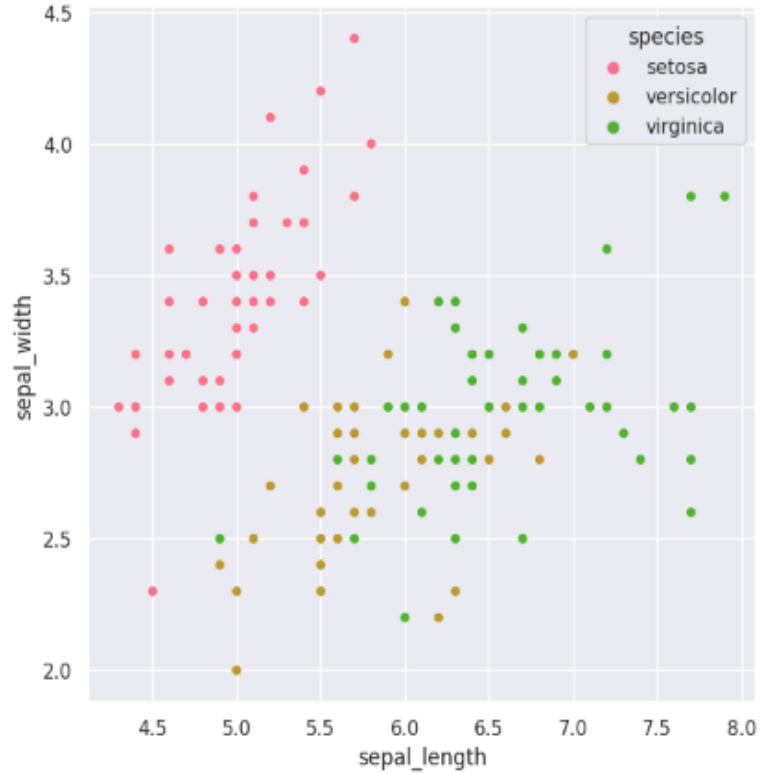

```
plt.figure(figsize=(7,7))
plt.plot(df_setosa['petal_length'], np.zeros_like(df_setosa['petal_length']), 'D')
plt.plot(df_virginica['petal_length'], np.zeros_like(df_virginica['petal_length']), 'D')
plt.plot(df_versicolor['petal_length'], np.zeros_like(df_versicolor['petal_length']), 'D')
plt.xlabel('Petal length')
plt.show()
```



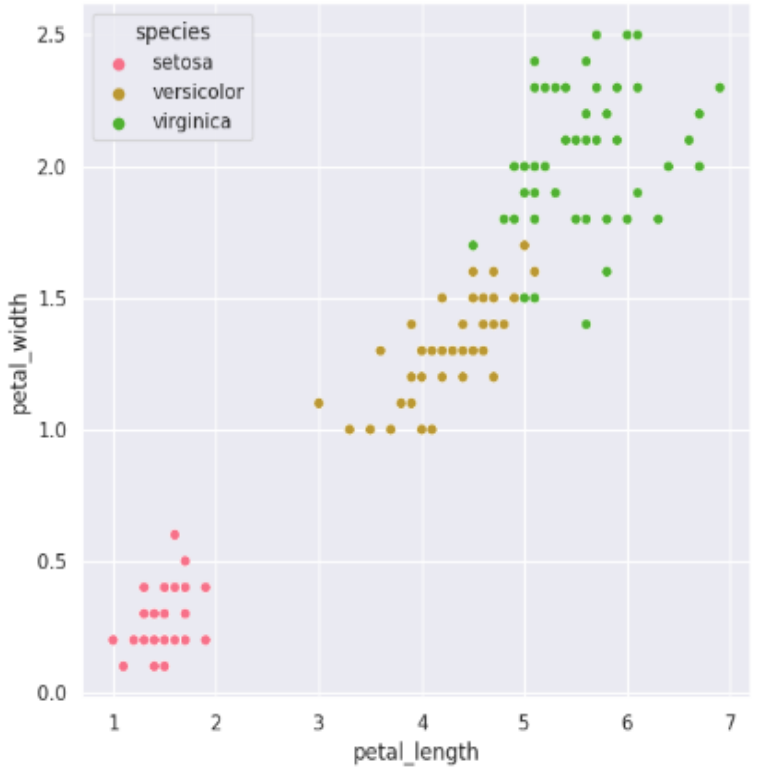
Data Analysis (3/6)

Bivariate Analysis

```
plt.figure(figsize=(7,7))
sns.scatterplot(df,x='sepal_length',y='sepal_width',hue='species')
plt.show()
```




```
plt.figure(figsize=(7,7))
sns.scatterplot(df,x='petal_length',y='petal_width',hue='species')
plt.show()
```

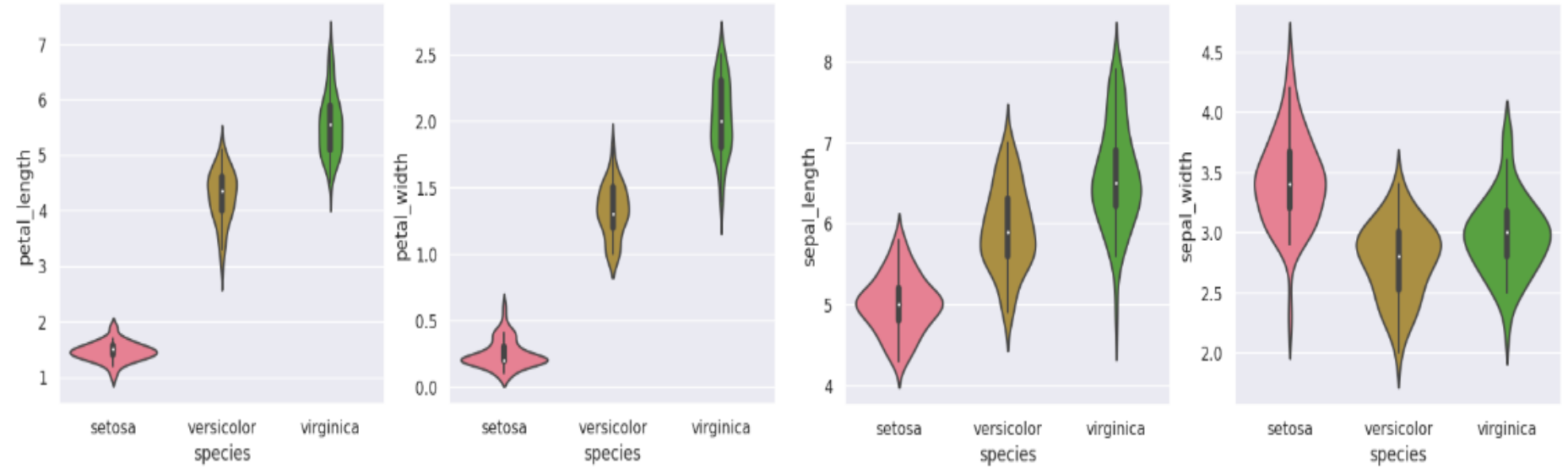


Data Analysis (4/6)

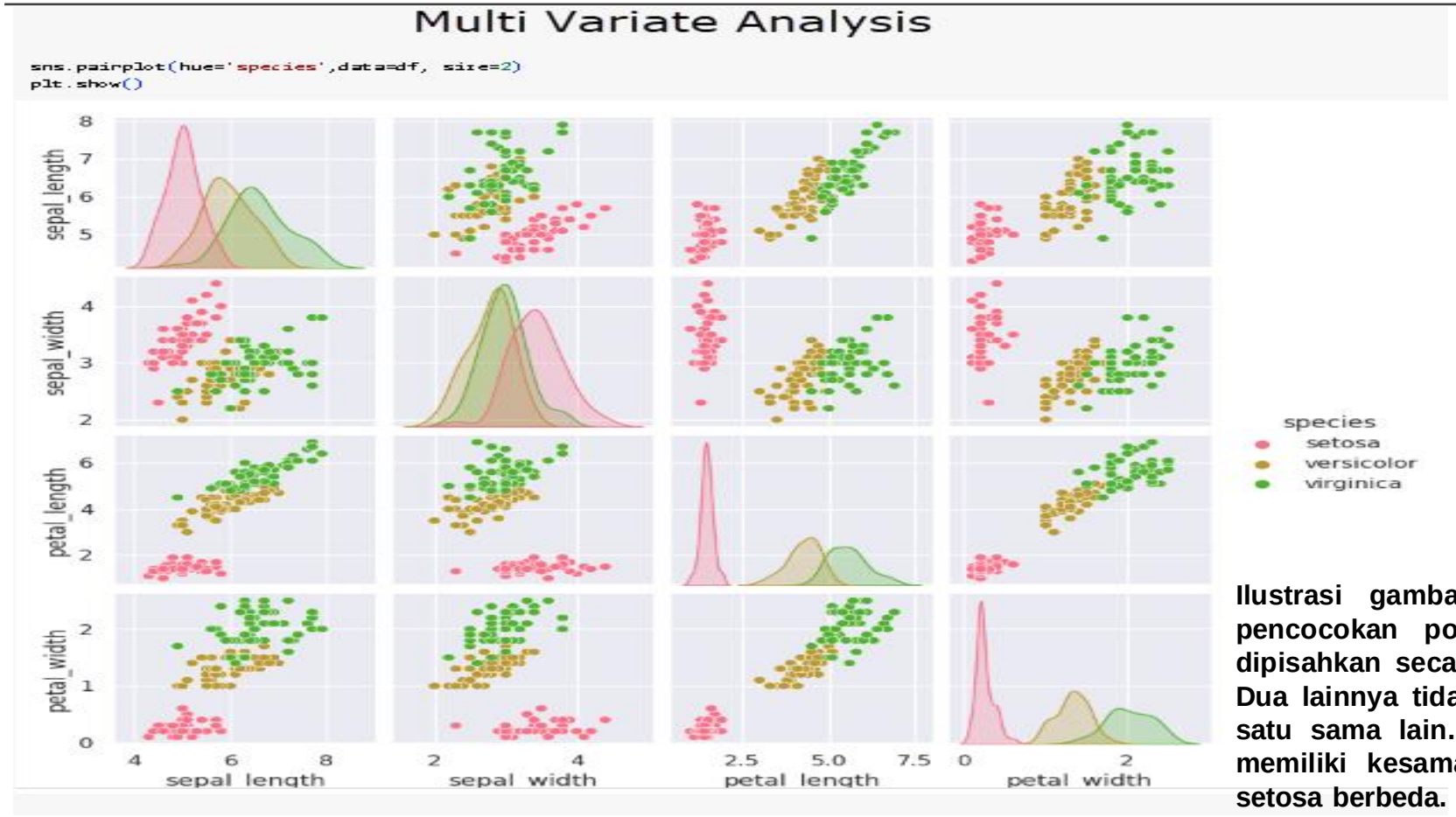
Bivariate Analysis

```
plt.figure(figsize=(10,10))
plt.subplot(2,2,1)
sns.violinplot(x='species',y='petal_length',data=df)
plt.subplot(2,2,2)
sns.violinplot(x='species',y='petal_width',data=df)
plt.subplot(2,2,3)
sns.violinplot(x='species',y='sepal_length',data=df)
plt.subplot(2,2,4)
sns.violinplot(x='species',y='sepal_width',data=df)
plt.show()
```

Ilustrasi gambar perbandingan spesies dan feature memberi gambaran yang lebih jelas, dimana spesies virginica secara keseluruhan lebih panjang dan lebar (kecuali pada sepal_width). Sedangkan spesies setosa secara keseluruhan lebih pendek dan kecil (kecuali pada sepal_width).



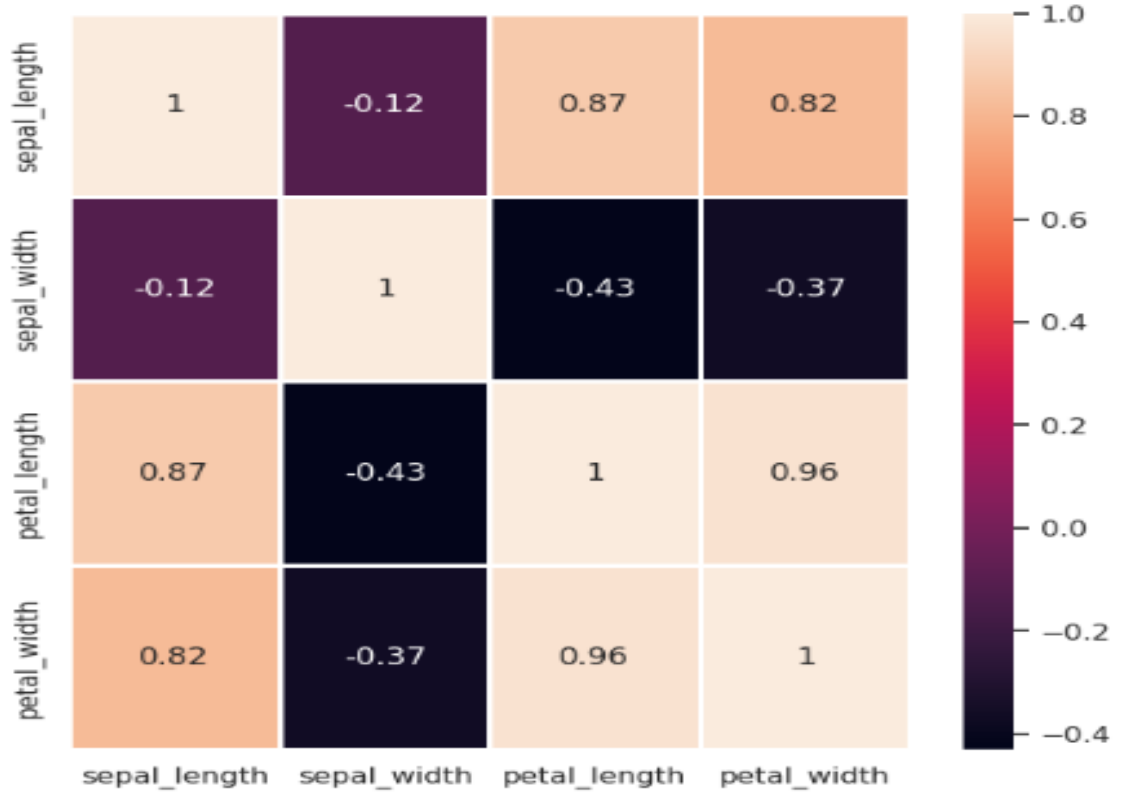
Data Analysis (5/6)



Ilustrasi gambar plot sangat berguna dalam pencocokan pola, dengan satu bunga dapat dipisahkan secara linier dari dua bunga lainnya. Dua lainnya tidak dapat dipisahkan secara linier satu sama lain. Iris-versicolor dan Iris-virginica memiliki kesamaan yang kuat, sedangkan Iris-setosa berbeda.

Data Analysis (6/6)

```
plt.figure(figsize=(7,7))
sns.heatmap(df.corr(),annot=True,linewidths=1)
plt.show()
```



Sepal_width tidak mempunyai korelasi dengan Feature lainnya, sedangkan Petal_length and Petal_width mempunyai korelasi yang tinggi dibandingkan features lainnya.

Data Modelling (1/4)

Modelling

Train test split

```
[84] X = df.drop("species", axis=1)
     y = df["species"]

     np.random.seed(42)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Train the model

```
[85] X_train
```

	sepal_length	sepal_width	petal_length	petal_width
22	4.6	3.6	1.0	0.2
15	5.7	4.4	1.5	0.4
65	6.7	3.1	4.4	1.4
11	4.8	3.4	1.6	0.2
42	4.4	3.2	1.3	0.2
...	...	...	...	...
71	6.1	2.8	4.0	1.3
106	4.9	2.5	4.5	1.7
14	5.8	4.0	1.2	0.2
92	5.8	2.6	4.0	1.2
102	7.1	3.0	5.9	2.1

120 rows × 4 columns

Modelling diawali dengan melakukan Train Test.

Data Modelling (2/4)

```
# Put models in a dictionary

models = {
    "Logistic Regression": LogisticRegression(),
    "KNN": KNeighborsClassifier(),
    "Random Forest": RandomForestClassifier()
}


# Create a function to fit and score models
def fit_and_score(models, X_train, X_test, y_train, y_test):
    """
    Fits and evaluates given machine learning models.
    models : a dict of different Scikit-Learn machine learning models
    X_train : training data (no labels)
    X_test : testing data (no labels)
    y_train : training labels
    y_test : test labels
    """
    # Set random seed
    np.random.seed(42)
    # Make a dictionary to keep model scores
    model_scores = {}
    # Loop through models
    for name, model in models.items():
        # Fit the model to the data
        model.fit(X_train, y_train)
        # Evaluate the model and append its score to model_scores
        model_scores[name] = model.score(X_test, y_test)
    return model_scores
```

```
model_scores = fit_and_score(models, X_train, X_test, y_train, y_test)

model_scores
```

```
{'Logistic Regression': 1.0, 'KNN': 1.0, 'Random Forest': 1.0}
```

Menentukan Model yang dipilih untuk mendapatkan perbandingan model score terbaik.

Data Modelling (3/4)

Hyperparameter tuning with GridSearchCV

```
[88] grid_knn = { 'n_neighbors' : [5,7,9,11,13,15],
                  'weights' : ['uniform','distance'],
                  'metric' : ['minkowski','euclidean','manhattan']}
     grid_clf = {
         'bootstrap': [True],
         'max_depth': [80, 90, 100, 110],
         'max_features': [2, 3],
         'min_samples_leaf': [3, 4, 5],
         'min_samples_split': [8, 10, 12],
         'n_estimators': [100, 200, 300, 1000]
     }
```

```
[89] # Tune KNN

     np.random.seed(42)

     # Setup random hyperparameter search for KNN
     gs_knn = GridSearchCV(KNeighborsClassifier(),
                           param_grid=grid_knn,
                           cv=5,
                           verbose=True)

     # Fit random hyperparameter search model for KNN
     gs_knn.fit(X_train, y_train)
```

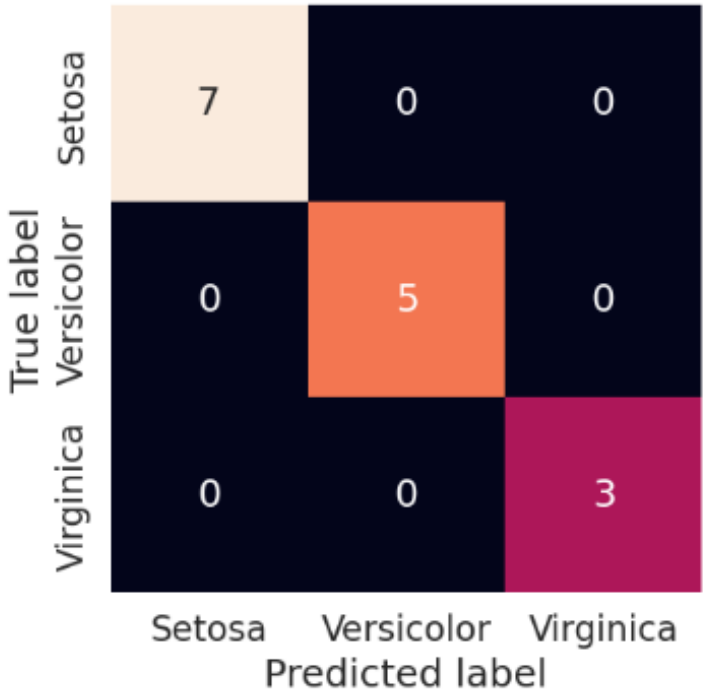
```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
```

GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier

Kemudian dilakukan Hyperparameter Tuning

Data Modelling (4/4)

```
# Visualizing Confusion matrix

sns.set(font_scale=1.5)


def plot_conf_mat(y_test, y_preds):
    """
    Plots a confusion matrix using Seaborn's heatmap()
    """
    fig, ax = plt.subplots(figsize=(5, 5))
    ax = sns.heatmap(confusion_matrix(y_test, y_preds), annot=True, cbar=False)
    plt.xlabel("Predicted label")
    plt.ylabel("True label")

    fig.canvas.draw()
    labels = ['Setosa',"Versicolor","Virginica"]

    ax.set_xticklabels(labels)
    ax.set_yticklabels(labels)

plot_conf_mat(y_test, y_preds)
```




```
[ ] scores = cross_val_score(clf, X, y, cv=5)

[ ] print('Model accuracy: ', np.mean(scores))

    Model accuracy:  0.9533333333333334
```

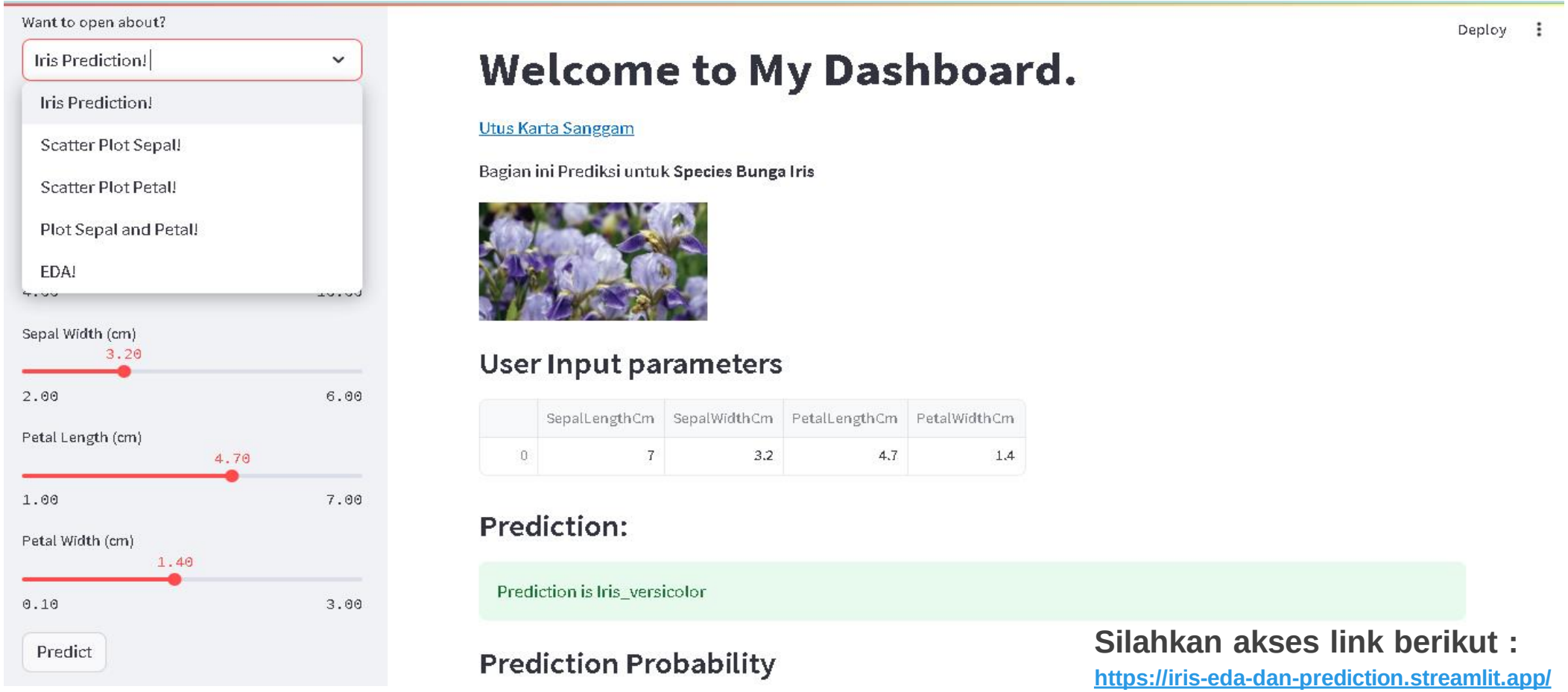
Hasil yang didapat Random Forest Classifier is the best model dengan acccuracy = 0.9533 (95.33 %).

Conclusion

Untuk menjawab bagaimana menentukan jenis spesies bunga Iris, kita dapat melakukan:

1. Data Quality dengan mengecek terlebih dahulu data yang kita miliki.
2. Data Analysis untuk mendapatkan overview data.
3. Dari hasil proses didapat 2 variabel/features yaitu Sepal dan Petal untuk mengidentifikasikan jenis spesies : Iris-setosa, Iris-versicolor atau Iris-verginica
4. Data Modelling dan Hyperparameter Tuning untuk memilih Random Forest sebagai model terbaik dari 3 model yang ditentukan (Logistic Regression, KNN/K-Nearest Neighbor dan Random Forest).

Membuat Dashboard



Silahkan akses link berikut :

https://iris-eda-dan-prediction.streamlit.app/

My Profile

Working Experience.

Data Analyst Internship in PT. Xeratic
June-July 2023

Career Transition to learn about Data
Sept 2023 – Present :

- Bootcamp Machine Learning & AI for Beginner - period July 1 - Aug 26, 2023 by DQLab.
- Data Engineer Course (Digital Talent Scholarship PROA 2023 Batch 1 period Mar-Apr, 2023) by DQLab.
- Bootcamp Data Analyst with SQL and Python period Jan-Mar, 2003 by DQLab
- Google Data Analytics Course (Digital Talent Scholarship PROA 2022-Batch 4 period Sep-Nov, 2022) by Cousera & Dicoding

www.linkedin.com/in/utusks01