Applying Machine Learning On

Iris Dataset

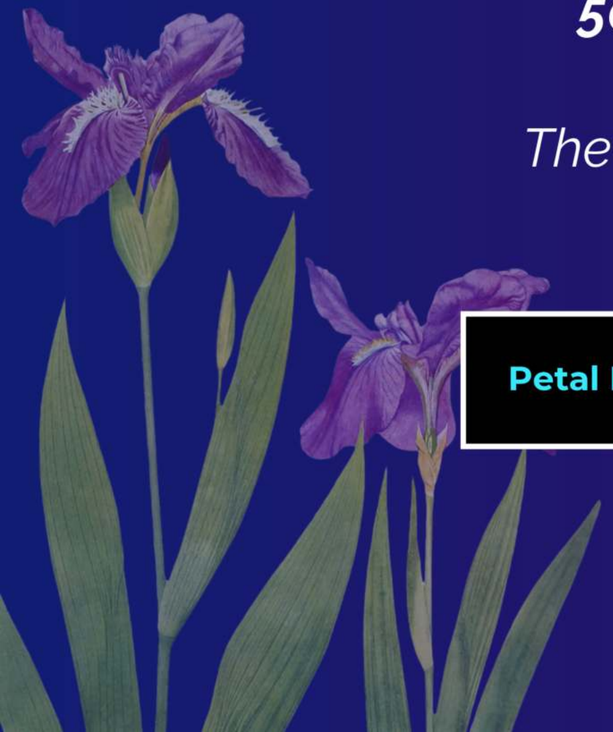
Group 1

Toh Kar Ming (21091137) Ang Yen Ping (22023170) Tung Qi Yong (22022750) Syehrran A/L Arulsamy (22000608)

Overview of Iris Flower Dataset

Ronald Fisher introduced this multivariate dataset in 1936. It is available in Kaggle, where we took ours. The dataset contains ***5 features*** and ***50 samples for each species***, ***summing up to 150 records***.

The species including Iris Setosa, Iris virginica, and Iris versicolor.



Petal Length	Petal Width	Setal Length	Setal Width	Species

Overview of Iris Flower Dataset

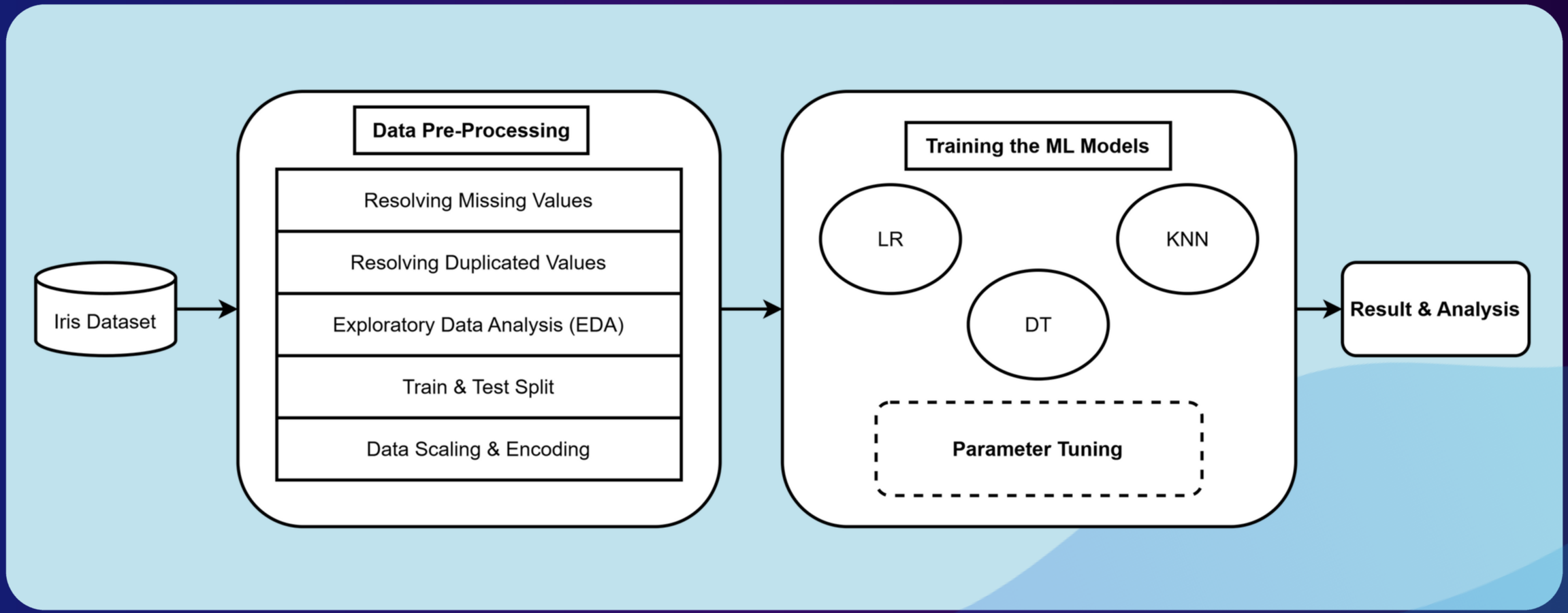
```
iris['species'].value_counts()
```

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

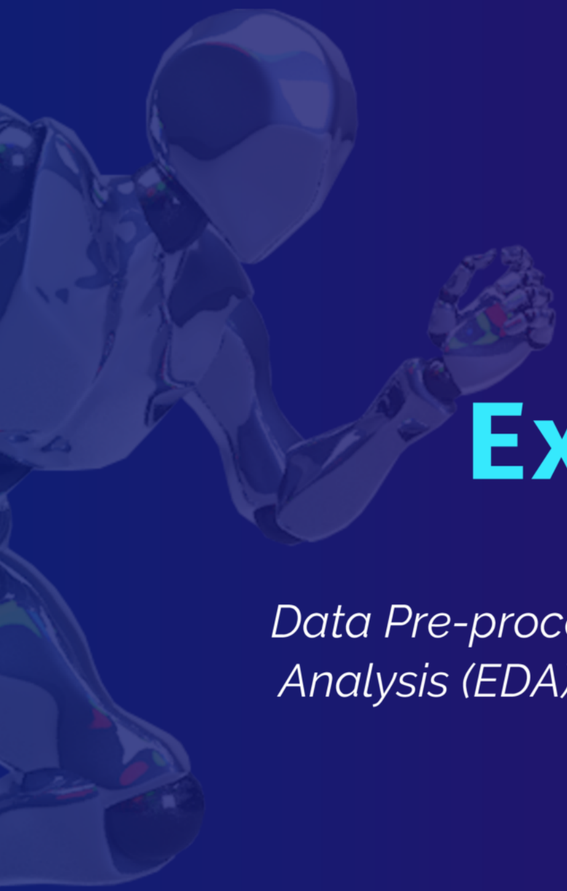
```
iris.info()
```

```
<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)
```

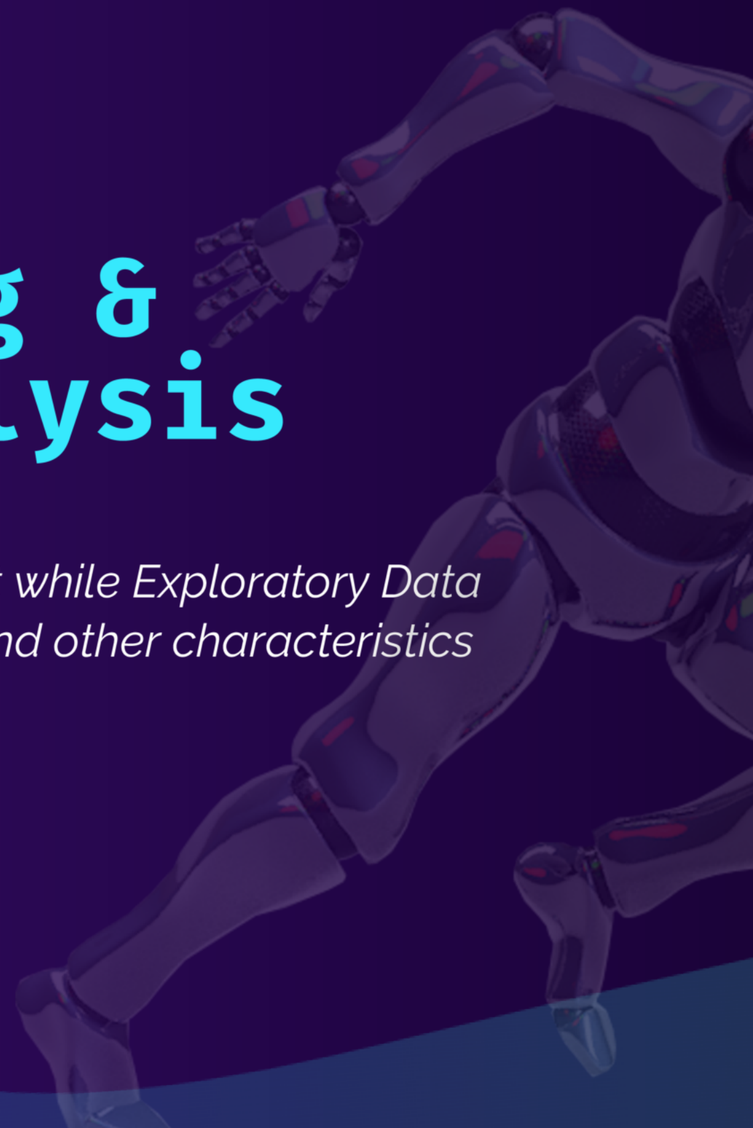
Methodology



Preprocessing & Analyzing the Dataset



Data Pre-processing & Exploratory Data Analysis



Data Pre-processing such as resolving null values and scaling is carried out while Exploratory Data Analysis (EDA) is used to analyzing data sets to identify patterns, outliers, and other characteristics

Checking Dataset - 1

Null value

None

```
iris.isnull().sum()
```

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

Checking Dataset - 2

Data Duplication

```
print(True in iris.duplicated(keep=False).values)
```

```
True
```

Check if any

```
iris.duplicated(keep=False)
print(f"Number of duplicated records:{iris.duplicated(keep=False).sum()}")
```

```
Number of duplicated records:5
```

How many in total?

Checking Dataset – 2

Data Duplication

keep = “first”
show all except the first occurrence

keep = “last”
show all except the last occurrence

```
[193]:
print(iris[iris.duplicated(keep="first")])
     sepal_length  sepal_width  petal_length  petal_width         species
34            4.9          3.1           1.5          0.1     Iris-setosa
37            4.9          3.1           1.5          0.1     Iris-setosa
142           5.8          2.7           5.1          1.9  Iris-virginica

[194]:
print(iris[iris.duplicated(keep="last")])
     sepal_length  sepal_width  petal_length  petal_width         species
9             4.9          3.1           1.5          0.1     Iris-setosa
34            4.9          3.1           1.5          0.1     Iris-setosa
101           5.8          2.7           5.1          1.9  Iris-virginica

[195]:
print(iris[iris.duplicated(keep=False)])
     sepal_length  sepal_width  petal_length  petal_width         species
9             4.9          3.1           1.5          0.1     Iris-setosa
34            4.9          3.1           1.5          0.1     Iris-setosa
37            4.9          3.1           1.5          0.1     Iris-setosa
101           5.8          2.7           5.1          1.9  Iris-virginica
142           5.8          2.7           5.1          1.9  Iris-virginica

[196]:
iris_dropDuplicates = iris.drop_duplicates()
print(True in iris_dropDuplicates.duplicated(keep=False).values)
False
```

Checking Dataset – 2

Data Duplication

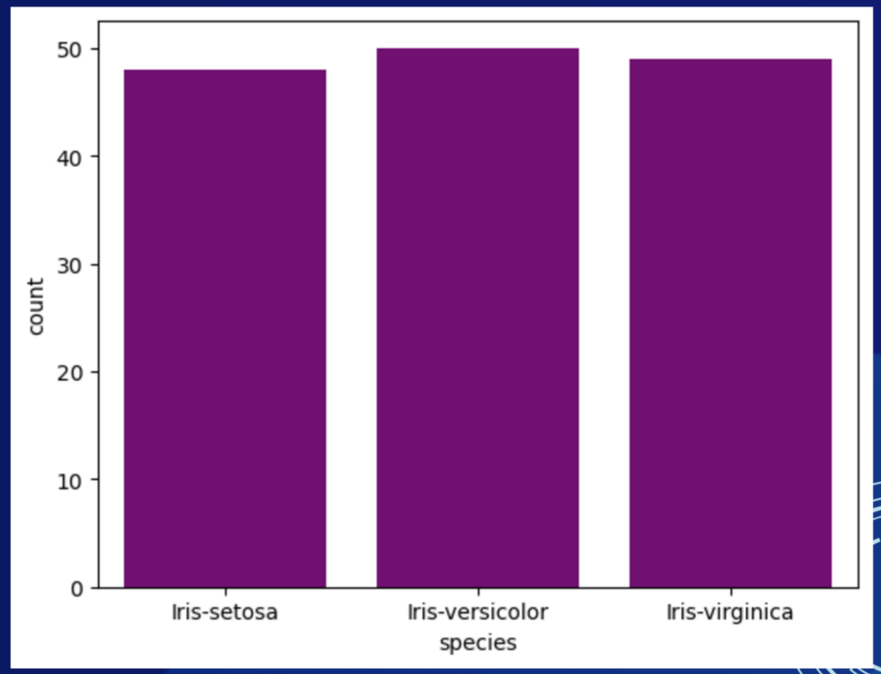
Drop duplicated records ->
check amount of records left

```
iris_dropDuplicates = iris.drop_duplicates()
print(True in iris_dropDuplicates.duplicated(keep=False).values)
```

```
False
```

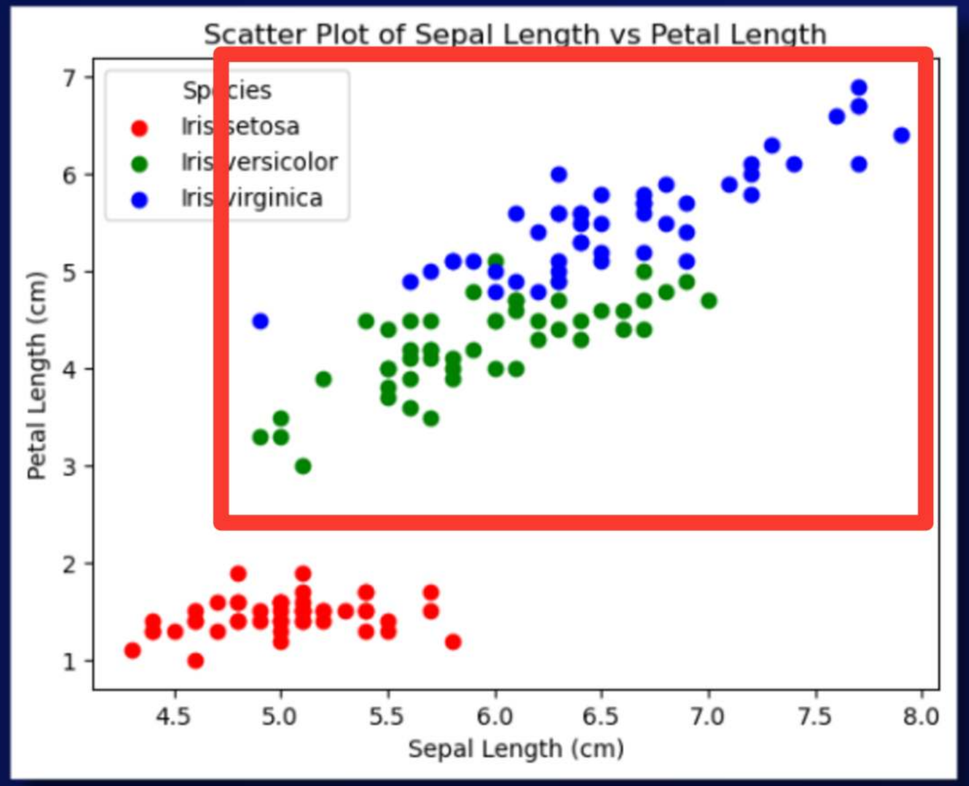
Dataset = 147 records

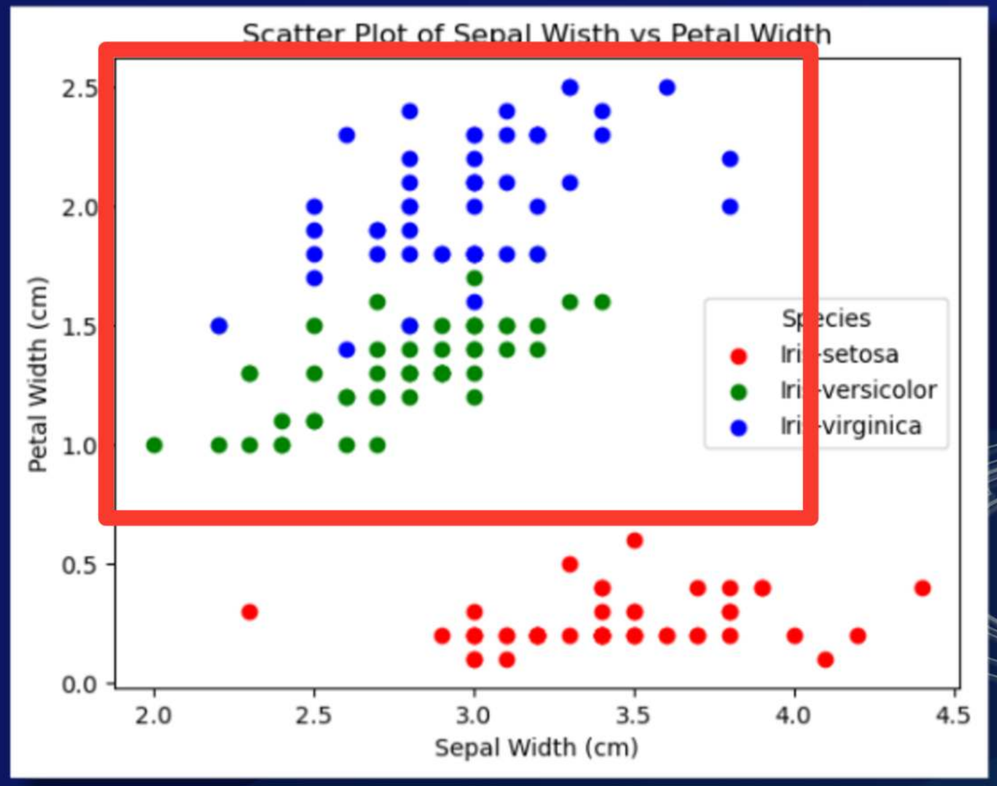
species	
Iris-versicolor	50
Iris-virginica	49
Iris-setosa	48



Gaining insight from dataset

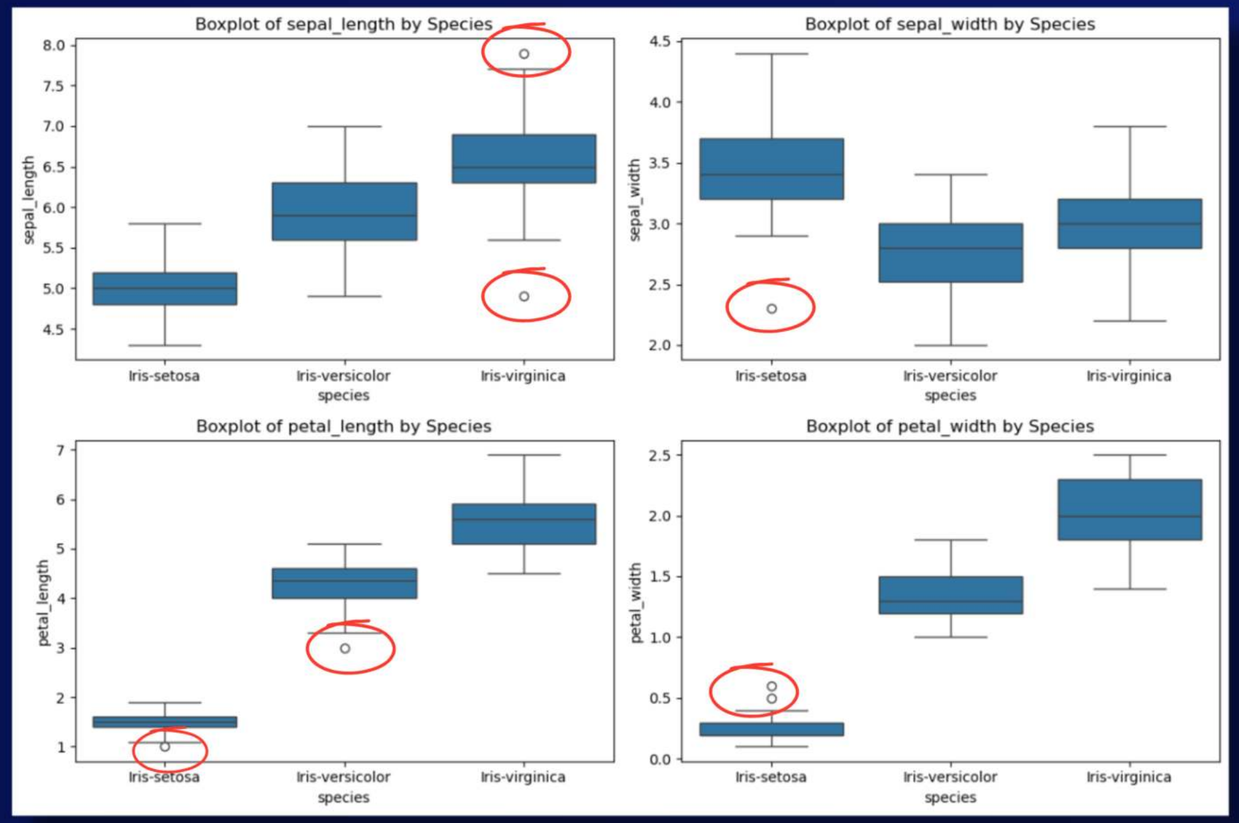
Distribution and Pattern





Gaining insight from dataset

Distribution and Pattern



7 outliers

Gaining insight from dataset

Encoding Target Values [species]

Sentosa = 0, Versicolor = 1,Virginica = 2

```
encoder = LabelEncoder()
iris_encoding = iris_dropDuplicates.copy()

# save target name before encoding for result analysis later on
target_names = iris_encoding['species'].unique()

# Now encode the species column
iris_encoding['species_encoded'] = encoder.fit_transform(iris_encoding['species'])
iris_encoded = iris_encoding.drop(columns=['species'])
iris_encoded
```

Gaining insight from dataset

Correlation Among Features

```
correlation_matrix = iris_encoded.corr()
```

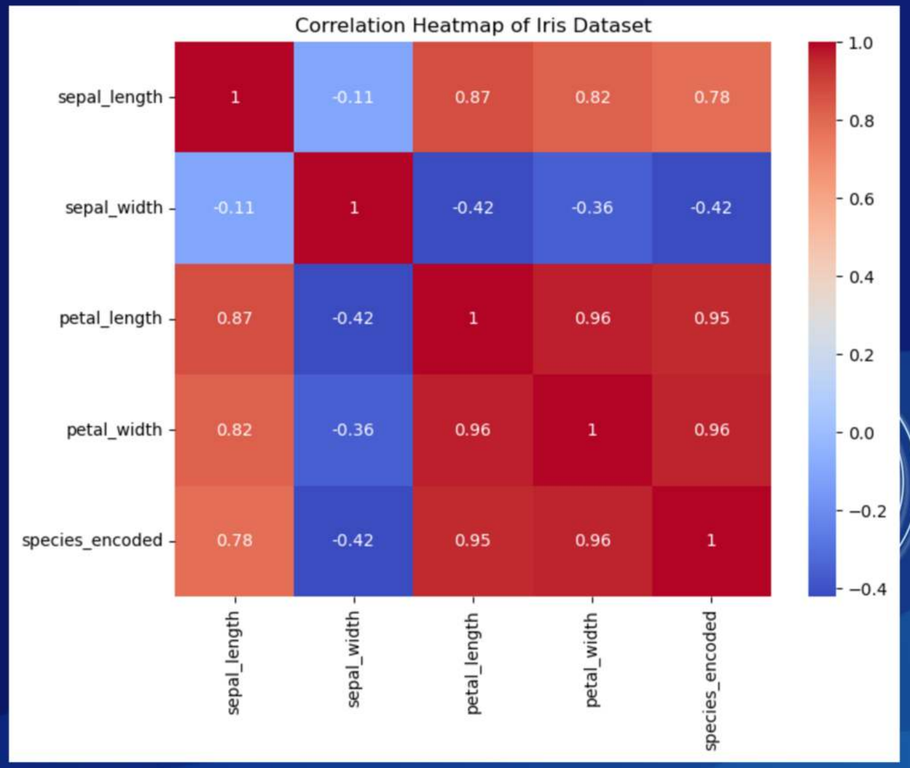
the correlation analysis includes species to show the interesting relationship among features.

NEGATIVE CORRELATION

between sepal_width and species

HIGHLY CORRELATED

petal_length, petal_width



Prepare data for modelling

Data Spliting

```
from sklearn.model_selection import train_test_split
X = iris_encoded.drop(columns=['species_encoded'])
y = iris_encoded['species_encoded']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Data Scaling / Normalisation

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Applying Algorithms to the Dataset

Model Training and Result Output

Using Logistic Regression (LR), K-Nearest Neighbors (KNN) and Decision Tree (DT)

Logistic Regression (LR)

WHAT:

- ML method that performs classification tasks
- Predict based on probability of a binary outcome

HOW:

LR can actually handle multiclass classification problems through the default One-vs-Rest (OvR) approach. The three classifiers of Iris dataset will be trained as follows:

- Classifier 1: setosa vs. (versicolor, virginica) = probability that the data point is setosa
- Classifier 2: versicolor vs. (setosa, virginica) = probability that the data point is versicolor
- Classifier 3: virginica vs. (setosa, versicolor) = probability that the data point is virginica

WHY:

LR assumes linear decision boundaries, which aligns well with Iris dataset simple feature relationships, making distinguishment between classes easier

Logistic Regression (LR)

1

```
# initializing LR model
lr = LogisticRegression (
    C= 10,
    max_iter = 100,
    random_state=42
)
```

✓ 0.0s

```
# training the LR model based on the train set
lr.fit(X_train_scaled, y_train)
```

✓ 0.0s

```
LogisticRegression
LogisticRegression(C=10, random_state=42)
```

```
# creating predictions of y based on x test
y_pred_lr = lr.predict(X_test_scaled)
```

✓ 0.0s

2

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred_lr)
print(f"Accuracy of Logistic Regression: {accuracy*100:.3f}")

# Detailed classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_lr, digits=4))
```

Finding Optimal C and Max Iterations

3

```
# Define parameter grid
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],
    'max_iter': [100, 200, 500, 1000]
}

# Logistic Regression model
lr_test = LogisticRegression()

# Grid search
grid_search = GridSearchCV(lr_test, param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters found: ", grid_search.best_params_)
```

✓ 1.0s

```
Best parameters found:  {'C': 10, 'max_iter': 100}
```

Evaluation

Accuracy of Logistic Regression: 97.778

Classification Report:

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	17
1	1.0000	0.9167	0.9565	12
2	0.9412	1.0000	0.9697	16
accuracy			0.9778	45
macro avg	0.9804	0.9722	0.9754	45
weighted avg	0.9791	0.9778	0.9776	45

Accuracy: 97.78%, Precision: 98.04%, Recall: 97.22%, F1-Score: 97.54%

Decision Tree

WHAT:

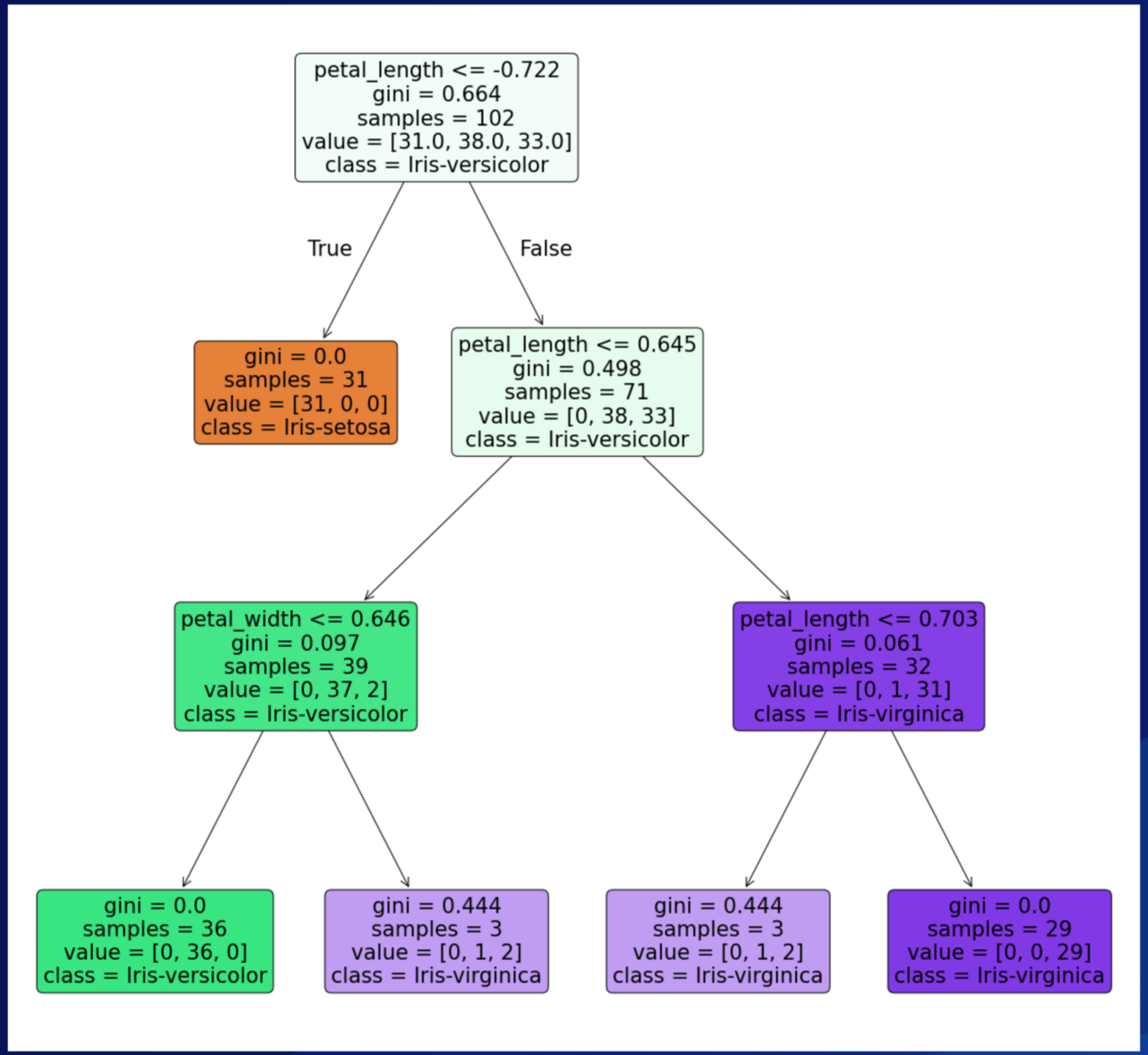
- Supervised machine learning algorithm.
- Splits data into branches using decision rules.
- Can be used for classification or regression.
- IRIS dataset Classification

WHY:

- Easy to interpret and visualise.
- Handles both numerical and categorical data.
- Performs well on small datasets like Iris.

Model Training

```
# initializing DTC model
dtc = DecisionTreeClassifier(criterion='gini', random_state=42, max_depth=3)
# training the DTC model based on the train set
dtc.fit(X_train_scaled, y_train)
# creating predictions of y based on x test
y_pred_dtc = dtc.predict(X_test_scaled)
```


petal_length <= -0.722
gini = 0.664
samples = 102
value = [31.0, 38.0, 33.0]
class = Iris-versicolor
True
False
gini = 0.0
samples = 31
value = [31, 0, 0]
class = Iris-setosa
petal_length <= 0.645
gini = 0.498
samples = 71
value = [0, 38, 33]
class = Iris-versicolor
petal_width <= 0.646
gini = 0.097
samples = 39
value = [0, 37, 2]
class = Iris-versicolor
petal_length <= 0.703
gini = 0.061
samples = 32
value = [0, 1, 31]
class = Iris-virginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = Iris-versicolor
gini = 0.444
samples = 3
value = [0, 1, 2]
class = Iris-virginica
gini = 0.444
samples = 3
value = [0, 1, 2]
class = Iris-virginica
gini = 0.0
samples = 29
value = [0, 0, 29]
class = Iris-virginica

Evaluation

```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred_dtc)
print(f"Accuracy of Classification Decision Tree: {accuracy*100:.3f}")

# Detailed classification report
print("\nClassification Report: ")
print(classification_report(y_test, y_pred_dtc))
```

```
Accuracy of Classification Decision Tree: 97.778

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000        17
           1     1.0000    0.9167    0.9565        12
           2     0.9412    1.0000    0.9697        16

    accuracy                         0.9778        45
   macro avg     0.9804    0.9722    0.9754        45
weighted avg     0.9791    0.9778    0.9776        45
```

Parameter Tuning

Result: max_depth 3 is the best.

- High Accuracy
- Avoid Overfitting

```python
# Parameter Tuning
max_depths =  [1, 2, 3, 5, 7, 20]
training_accuracy = []
testing_accuracy = []
plt.figure(figsize=(5, 4))

for i, md in enumerate(max_depths):
    dtc = DecisionTreeClassifier(criterion='gini', random_state=42, max_depth=md)
    dtc.fit(X_train_scaled, y_train)

    train_score = dtc.score(X_train_scaled, y_train)
    test_score = dtc.score(X_test_scaled, y_test)

    training_accuracy.append(train_score)
    testing_accuracy.append(test_score)

    print(f"Max Depth: {md}")
    print(f"Train Score: {train_score:.2f} | Test Score: {test_score:.2f}\n")
```

```
Max Depth: 1
Train Score: 0.68 | Test Score: 0.64

Max Depth: 2
Train Score: 0.97 | Test Score: 0.89

Max Depth: 3
Train Score: 0.98 | Test Score: 0.98

Max Depth: 5
Train Score: 1.00 | Test Score: 0.93

Max Depth: 7
Train Score: 1.00 | Test Score: 0.93

Max Depth: 20
Train Score: 1.00 | Test Score: 0.93
```

K-Nearest Neighbors

WHAT:

- Supervised machine learning algorithm.
- Assigns class based on the most frequent label among the k nearest neighbors.
- Uses a distance metric (e.g., Euclidean) to measure proximity.
- Applied to the Iris dataset for multi-class classification.

WHY:

- Easy to implement and interpret.
- Non-parametric: No assumptions about data distribution.
- Balances robustness and accuracy with optimal k-value tuning.

Model Training

```
# initializing KNN model
knn = KNeighborsClassifier(n_neighbors=1)
```

```
# Training the KNN model based on the train set
knn.fit(X_train_scaled, y_train)
```

KNeighborsClassifier

```
KNeighborsClassifier(n_neighbors=1)
```

```
# Creating predictions of y based on X_test
y_pred_knn = knn.predict(X_test_scaled)
```

Evaluation

```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred_knn)
print(f"Accuracy of K-Nearest Neighbors: {accuracy * 100:.3f}")

# Detailed classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_knn, digits=4))
```

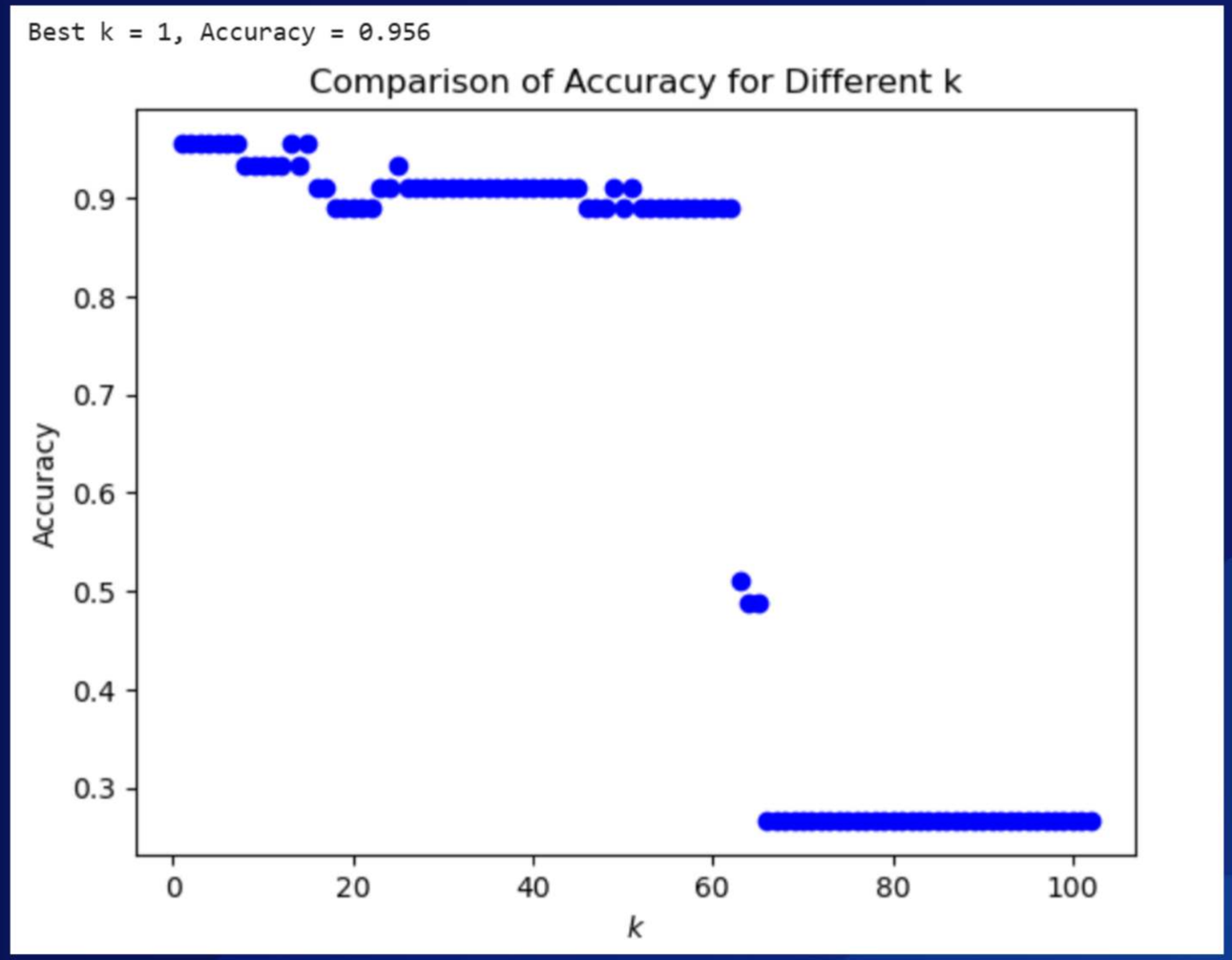
```
Accuracy of K-Nearest Neighbors: 93.333

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000        17
           1     0.9091    0.8333    0.8696        12
           2     0.8824    0.9375    0.9091        16

    accuracy                         0.9333        45
   macro avg     0.9305    0.9236    0.9262        45
weighted avg     0.9339    0.9333    0.9329        45
```

Comparison of Accuracy for Different K Values



Concluding the Study

Conclusion

Analysing Results & Future Improvement Suggestions to this Study

Result & Discussion

Models	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	97.78	98.04	97.22	97.54
K-Nearest Neighbours	93.33	93.05	92.36	92.62
Decision Tree	97.78	98.04	97.22	97.54

LR & DT EXPLANATION:

LR & DT had the same and best performance when compared to DT

This is because Iris dataset has **relatively well-separated classes**, especially between Iris-setosa and the other two species.

Our DT is probably relatively relatively simple or shallow, causing **similar decision boundaries** as compared to **linear decision boundaries** by LR. In other words, the **linear decision boundaries ends up being emulated by simple decision trees**.

Future Improvements

Exploring a Larger Iris Dataset

- Benefits
  - Improved Model Performance
  - Higher Accuracy

- Limitation of Small Dataset:
  - Small datasets may not capture all underlying relationships, leading to less accurate predictions and potential overfitting.

Application of Ensemble Methods

- A technique that combines the predictions of multiple models to produce a final prediction.
- Method:
  - Bagging
  - Boosting
  - Stacking
- Benefits:
  - To reduce overfitting
  - To improve performance metric

Thank You!