Task 2

You are given a dataset containing information about iris flowers, including features such as sepal length, sepal width, petal length, and petal width. Your task is to build a classification model to predict the species of the iris flower.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

data= pd.read_csv(r"C:\Users\91965\Downloads\Iris (1).csv")

data.head(10)

   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species
0   1            5.1           3.5            1.4           0.2  Iris-
setosa
1   2            4.9           3.0            1.4           0.2  Iris-
setosa
2   3            4.7           3.2            1.3           0.2  Iris-
setosa
3   4            4.6           3.1            1.5           0.2  Iris-
setosa
4   5            5.0           3.6            1.4           0.2  Iris-
setosa
5   6            5.4           3.9            1.7           0.4  Iris-
setosa
6   7            4.6           3.4            1.4           0.3  Iris-
setosa
7   8            5.0           3.4            1.5           0.2  Iris-
setosa
8   9            4.4           2.9            1.4           0.2  Iris-
setosa
9  10            4.9           3.1            1.5           0.1  Iris-
setosa

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
data.isnull().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
#there are no missing values in the dataset, let's move to next steps.

data['Species'].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
dtype=object)
```

```
#There are three types of the species, let's encode the target
variable
#To encode, we will use label Encoder

le= LabelEncoder()

data['Species_encoded']= le.fit_transform(data['Species'])

data.head()
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species  \
0   1            5.1           3.5            1.4           0.2  Iris-
setosa
1   2            4.9           3.0            1.4           0.2  Iris-
setosa
2   3            4.7           3.2            1.3           0.2  Iris-
setosa
```

```
3   4            4.6           3.1           1.5           0.2  Iris-
setosa
4   5            5.0           3.6           1.4           0.2  Iris-
setosa

   Species_encoded
0                0
1                0
2                0
3                0
4                0
```

```
#Now split the dataset into train_test_Split

X = data.drop(columns=['Species', 'Species_encoded', 'Id'])
X
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0              5.1           3.5            1.4           0.2
1              4.9           3.0            1.4           0.2
2              4.7           3.2            1.3           0.2
3              4.6           3.1            1.5           0.2
4              5.0           3.6            1.4           0.2
..             ...           ...            ...           ...
145            6.7           3.0            5.2           2.3
146            6.3           2.5            5.0           1.9
147            6.5           3.0            5.2           2.0
148            6.2           3.4            5.4           2.3
149            5.9           3.0            5.1           1.8

[150 rows x 4 columns]
```

```
y= data['Species_encoded']
y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: Species_encoded, Length: 150, dtype: int32
```

```
# Train_Test_split
```

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

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((120, 4), (120,), (30, 4), (30,))
```

```
# Initialize models
models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC(),
    'KNN': KNeighborsClassifier()
}

results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[model_name] = accuracy
    print(f"{model_name}: Accuracy = {accuracy}")
```

```
Logistic Regression: Accuracy = 1.0
Random Forest: Accuracy = 1.0
SVM: Accuracy = 1.0
KNN: Accuracy = 1.0
```

```
  # Generate classification report
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
#Conclusion- Since we are already getting the max accuracy so there is
no room for improvement in this data.



#Let's check the model output by giving input

new_value= pd.DataFrame({
    'SepalLengthCm': [3],
    'SepalWidthCm': [3],
    'PetalLengthCm': [4],
    'PetalWidthCm': [2]
})

predictions = model.predict(new_value)
predictions
```

```
array([1])
```