Experiment No.4

Title:Implementation of KNN Classifier on appropriate dataset

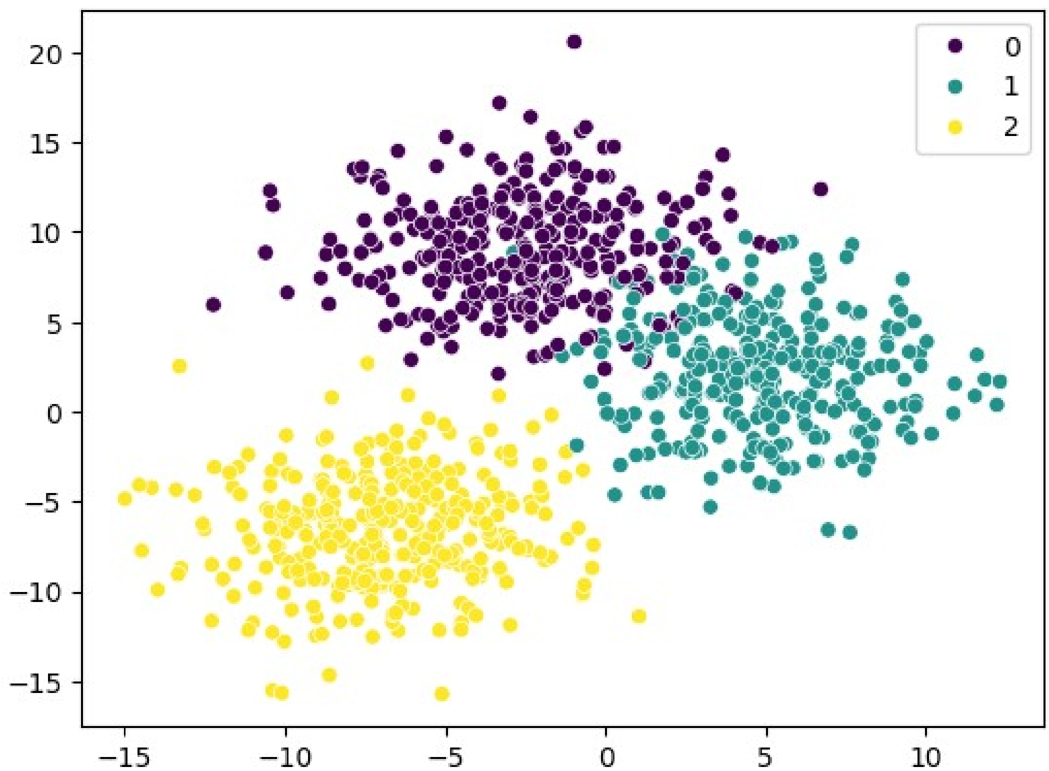
Part A:Implementation of Knn Classifier on Synthetic data

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.datasets import make_blobs
warnings.filterwarnings('ignore')
```

Creating and Visualizing Synthetic data

```
X,Y=make_blobs(n_samples=1000,n_features=2,centers=3,cluster_std=3,ran
dom_state=42)

sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=Y,palette='viridis')
plt.show()
```

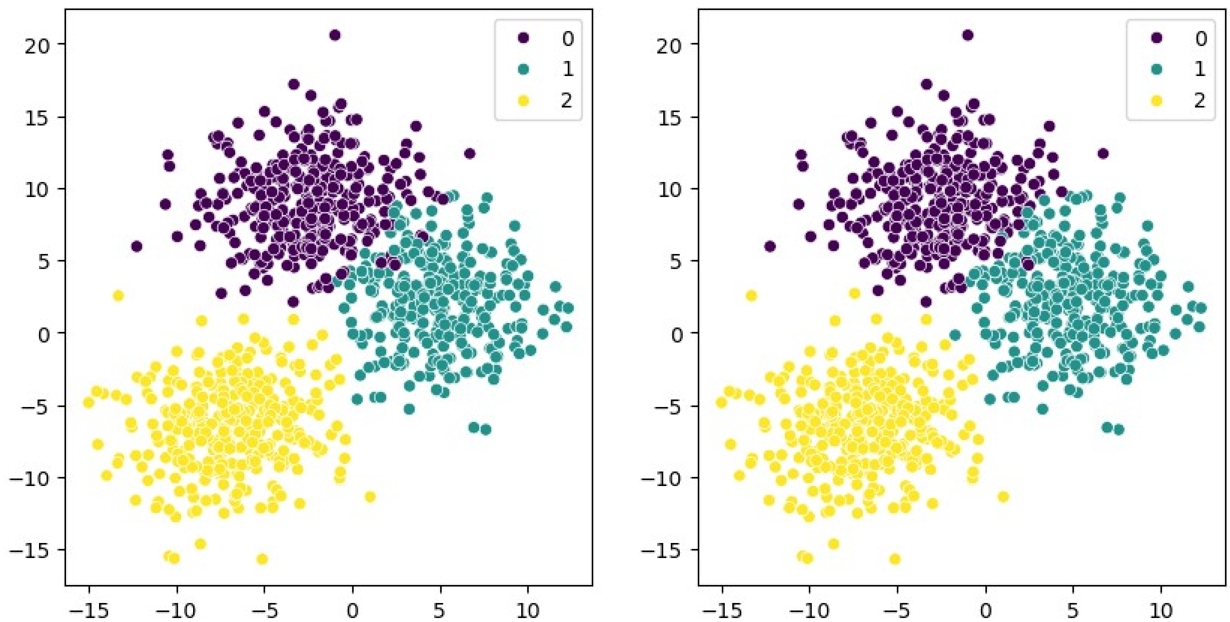


KNN Classifier Models

```
from sklearn.neighbors import KNeighborsClassifier
knn_classifier1=KNeighborsClassifier(n_neighbors=3)
knn_classifier2=KNeighborsClassifier(n_neighbors=5)

knn_classifier1.fit(X,Y)

KNeighborsClassifier(n_neighbors=3)

knn_classifier2.fit(X,Y)

KNeighborsClassifier()
```

Predicting the Labels

```
y_p1=knn_classifier1.predict(X)
y_p2=knn_classifier2.predict(X)

plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=y_p1,palette='viridis')

plt.subplot(1,2,2)
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=y_p2,palette='viridis')

<Axes: >
```



Performance Metrics

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report

print('Model with n_neighbors=3')
print('accuracy_score:',accuracy_score(Y,y_p1))
print('confusion_matrix: ')
print(confusion_matrix(Y,y_p1))
print('classification_report: ')
print(classification_report(Y,y_p1))

print('Model with n_neighbors=5')
print('accuracy_score:',accuracy_score(Y,y_p2))
print('confusion_matrix: ')
print(confusion_matrix(Y,y_p2))
print('classification_report: ')
print(classification_report(Y,y_p2))
```

```
Model with n_neighbors=3
accuracy_score: 0.979
confusion_matrix:
[[328   6   0]
 [ 13 319   1]
 [  1   0 332]]
classification_report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       334
           1       0.98      0.96      0.97       333
           2       1.00      1.00      1.00       333
```

```
    accuracy                           0.98      1000
   macro avg       0.98      0.98      0.98      1000
weighted avg       0.98      0.98      0.98      1000

Model with n_neighbors=5
accuracy_score: 0.968
confusion_matrix:
[[317  17   0]
 [ 13 319   1]
 [  0   1 332]]
classification_report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95       334
           1       0.95      0.96      0.95       333
           2       1.00      1.00      1.00       333

    accuracy                           0.97      1000
   macro avg       0.97      0.97      0.97      1000
weighted avg       0.97      0.97      0.97      1000
```

Part B:Implementation of Knn Classifier on a dataset

Load Penguins dataset from seaborn datasets

```
sns.get_dataset_names()
```

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
```

```
 'tips',
 'titanic']
```

```
df=sns.load_dataset('penguins')
df.head()
```

```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0   

1  Adelie  Torgersen            39.5           17.4              186.0   

2  Adelie  Torgersen            40.3           18.0              195.0   

3  Adelie  Torgersen             NaN            NaN                NaN   

4  Adelie  Torgersen            36.7           19.3              193.0   


   body_mass_g     sex  
0       3750.0    Male  
1       3800.0  Female  
2       3250.0  Female  
3          NaN     NaN  
4       3450.0  Female  
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

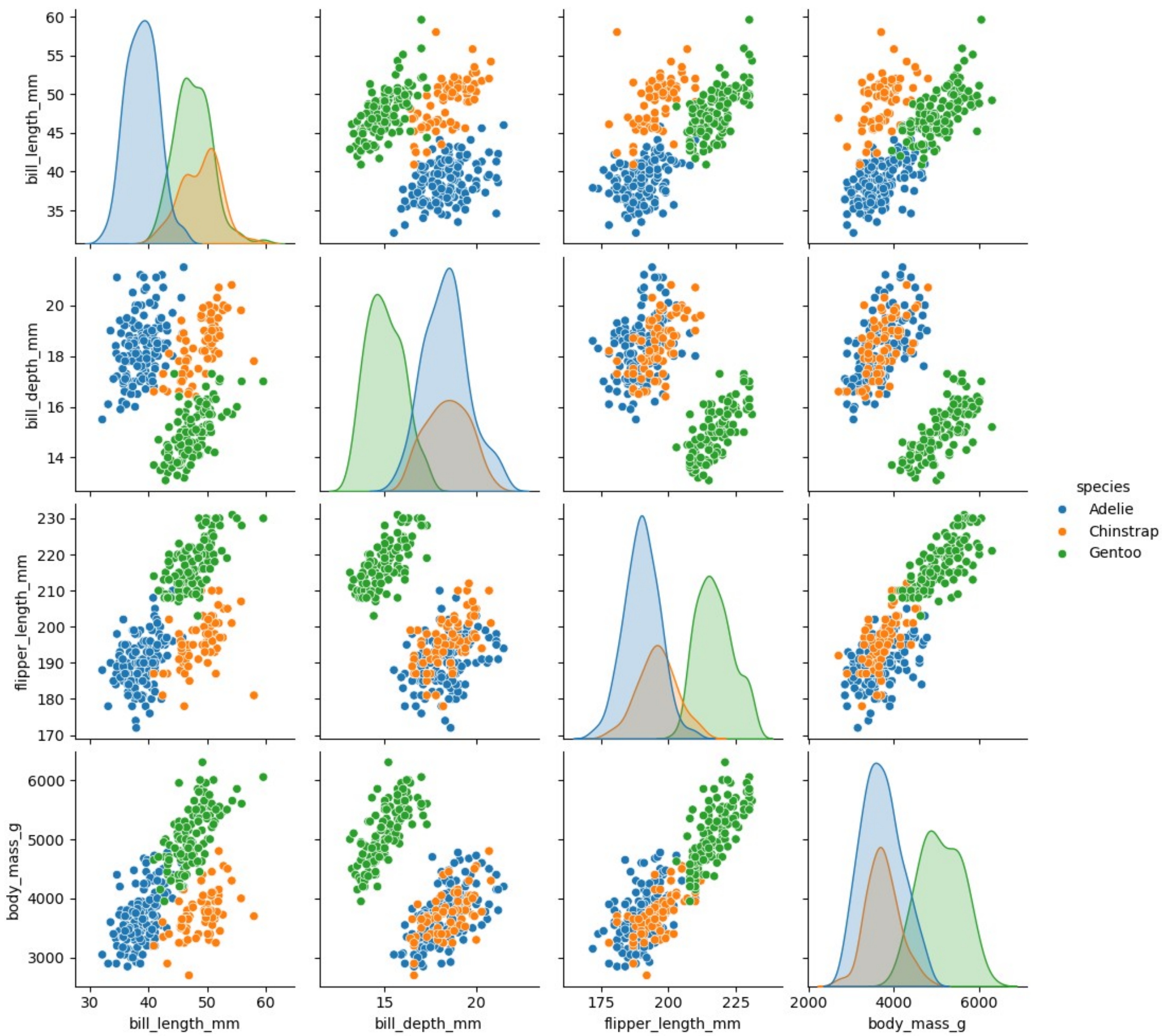
```
df.isnull().sum()#Checking for null values
```

```
species               0
island                0
bill_length_mm        2
bill_depth_mm         2
flipper_length_mm     2
body_mass_g           2
```

```
sex                    11
dtype: int64

df.dropna(inplace=True)#removing all null value rows

df.shape

(333, 7)
```

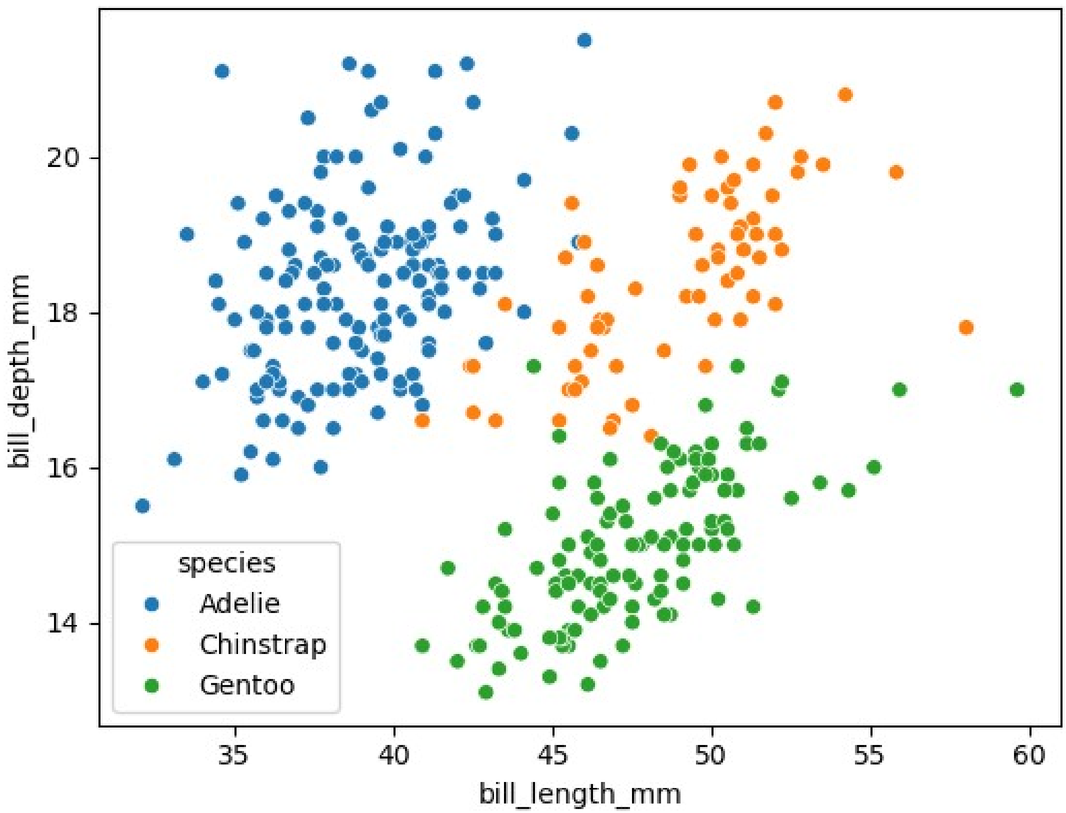
Visualizing the data for finding best features for classification

```
sns.pairplot(df,hue='species')

<seaborn.axisgrid.PairGrid at 0x743e0b04ea80>
```



Separating data into features and Labels/Target

```
X=df[['bill_length_mm','bill_depth_mm']]
Y=df['species']

sns.scatterplot(x='bill_length_mm',y='bill_depth_mm',hue='species',dat
a=df)

<Axes: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```



Splitting the data into train and test sets

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,rando
m_state=0)
```

KNN Classifier Model

```
from sklearn.neighbors import KNeighborsClassifier

knn_classifier=KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train,Y_train)

KNeighborsClassifier()
```

Predicting Penguin Species on test data

```
y_pred=knn_classifier.predict(X_test)
y_pred
```

```
array(['Adelie', 'Adelie', 'Gentoo', 'Adelie', 'Adelie', 'Adelie',
       'Gentoo', 'Gentoo', 'Gentoo', 'Chinstrap', 'Gentoo', 'Adelie',
       'Adelie', 'Chinstrap', 'Adelie', 'Adelie', 'Gentoo', 'Adelie',
       'Chinstrap', 'Adelie', 'Adelie', 'Adelie', 'Gentoo', 'Gentoo',
       'Gentoo', 'Gentoo', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
       'Adelie', 'Chinstrap', 'Adelie', 'Adelie', 'Adelie', 'Gentoo',
       'Chinstrap', 'Adelie', 'Chinstrap', 'Adelie', 'Gentoo',
'Gentoo',
       'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie',
       'Gentoo', 'Adelie', 'Adelie', 'Adelie', 'Gentoo', 'Chinstrap',
       'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', 'Gentoo',
       'Adelie', 'Chinstrap', 'Adelie', 'Gentoo', 'Adelie', 'Adelie',
       'Gentoo'], dtype=object)
```

Model Performance Evaluation

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
print('accuracy_score:',accuracy_score(Y_test,y_pred))
print('confusion_matrix: ')
print(confusion_matrix(Y_test,y_pred))
print('classification_report: ')
print(classification_report(Y_test,y_pred))
```

```
accuracy_score: 0.9402985074626866
confusion_matrix:
[[39  0  0]
 [ 2  7  1]
 [ 0  1 17]]
classification_report:
              precision    recall  f1-score   support

      Adelie       0.95      1.00      0.97        39
   Chinstrap       0.88      0.70      0.78        10
      Gentoo       0.94      0.94      0.94        18

    accuracy                           0.94        67
   macro avg       0.92      0.88      0.90        67
weighted avg       0.94      0.94      0.94        67
```

Conclusion:

K-Nearest Neighbors (KNN) Classifier

Merits of KNN Classifier

1. **Simplicity**:
   - KNN is easy to understand and implement, making it a great choice for beginners.
2. **No Training Phase**:
   - KNN is a lazy learner, meaning it does not require a training phase. It simply stores the training data and makes predictions based on that data.
3. **Flexibility**:
   - It can be used for both classification and regression tasks, providing versatility.
4. **Adaptability**:
   - KNN can work with any number of classes and does not make any assumptions about the underlying data distribution.
5. **Effectiveness with Large Datasets**:
   - It can perform well with large datasets where decision boundaries are complex.
6. **Local Decision Making**:
   - KNN uses local information to make decisions, making it robust to outliers.

Demerits of KNN Classifier

1. **Computational Complexity**:
   - KNN requires computing the distance between the test instance and all training samples, which can be time-consuming, especially with large datasets.
2. **Memory Intensive**:
   - Since KNN stores all the training data, it can consume a significant amount of memory.
3. **Sensitivity to Feature Scaling**:
   - The performance of KNN can be adversely affected by the scale of features. Features need to be normalized or standardized.
4. **Choice of K**:
   - Selecting the optimal number of neighbors (K) can be challenging. A small K can lead to noise, while a large K can smooth out important patterns.
5. **Curse of Dimensionality**:
   - As the number of dimensions increases, the distance between points becomes less meaningful, which can degrade performance.
6. **Imbalanced Data**:
   - KNN can be biased towards the majority class in imbalanced datasets, potentially leading to poor classification performance for minority classes.