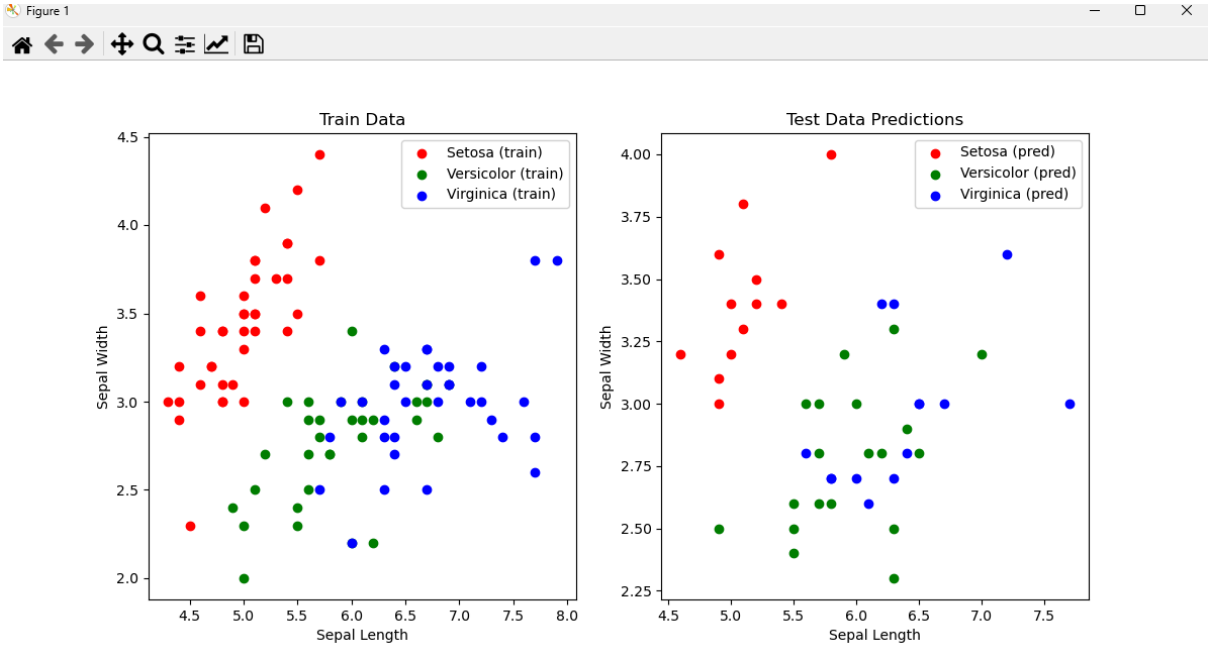
Augusto César Muniz Lopes
Trabalho 01
Tipo 9 - Classificação não-paramétrica (Estimador de Kernel e K-NN)

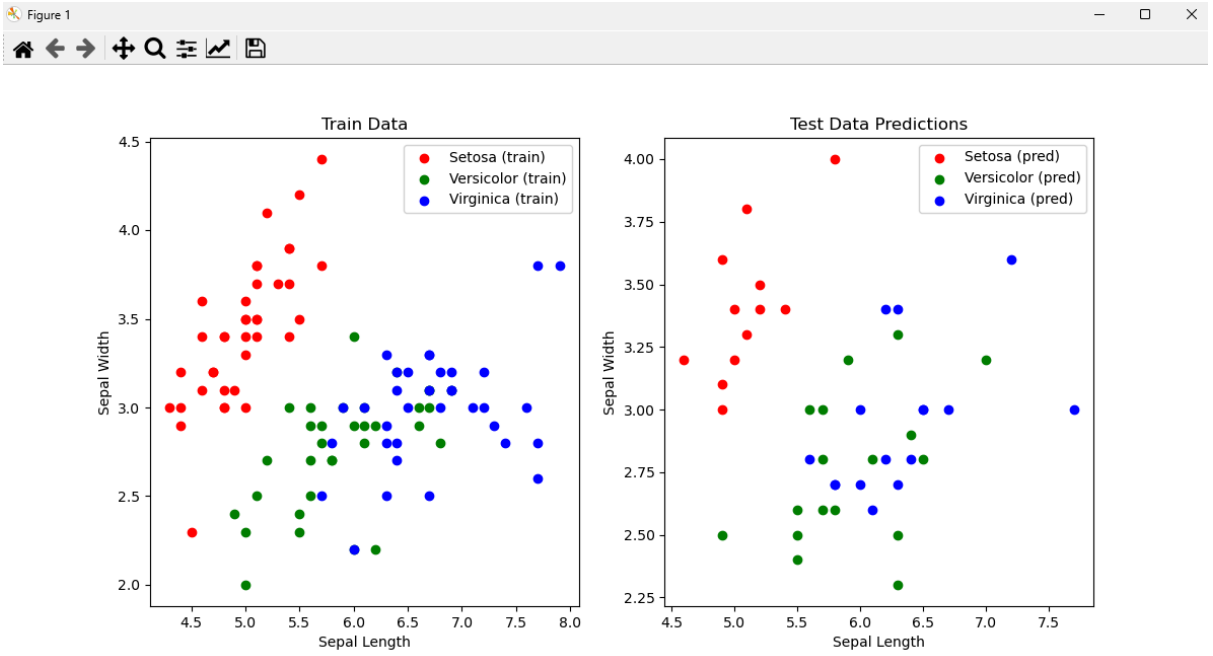
- Os dados utilizados em ambos os algoritmos foram do Dataset *Iris* disponível na biblioteca do “sklearn”. Para testar utilizei a ideia que vimos em sala de aula de dividir as amostras em 70% treinamento e 30% teste como demonstrado na imagem abaixo.

```
from sklearn import datasets
import math
from sklearn.model_selection import train_test_split # dividir um conjunto de dados em conjuntos de treinamento e teste.
from sklearn import metrics # fornece funções para calcular métricas de avaliação de modelos, como a acurácia, que será usada para avaliar o desempenho do classificador.

# Load data
iris = datasets.load_iris() # 150 amostras
# quatro características (X): comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala
# classes (espécies) das flores [0 = Setosa, 1 = Versicolor e 2 = Virginica] (Y)

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3, random_state=109)
```

- Em ambos os algoritmos utilizei as variáveis globais “X_train”, e “y_train” para os pontos de dados observados (amostra) e “X_test” e “y_test” para as observações de teste. No final dos testes, os valores de predição são armazenados em uma lista chamada “predict”.
    - Nos estimador de Kernel “h” é minha bandwidth que escolhi testando alguns valores e “d” refere-se dimensionalidade para *Multivariate Data*, ou seja, as 4 características do meu dataset (comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala). Além disso, utilizei algumas variáveis auxiliares uma para armazenar a soma de cada classe (sum_c0, sum_c1 e sum_c2) utilizada no cálculo da minha função kernel e outra para armazenar a quantidade de amostras de cada classe (N_c0, N_c1 e N_c2). Por fim, ´”u“ é meu argumento, calculado pela distância normalizada entre minha amostra e minha observação.
    - No classificador K-NN a variável “k” representa o número de vizinhos próximos que vou escolher para avaliar suas classes. A lista “dist” vai servir como auxiliar para descobrir a proximidade dos pontos, enquanto “min” e “index” me auxiliam para descobrir quais são os menores valores presentes na lista de distância. Por fim, “ki_c0”, “ki_c1” e “ki_c2” são a quantidade de vezes que cada classe repete.

- Gráficos plotados em função do comprimento e largura da sépala abaixo (Estimador de Kernel e K-NN, respectivamente)



- Para o estimador de Kernel utilizei como configuração ideal h = 0,5 onde obtive os resultados abaixo

```
Accuracy: 0.9111111111111111
Precision: 0.9235588972431078
Recall: 0.9178921568627452
F1-Score:                precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.84      0.94      0.89        17
           2       0.93      0.81      0.87        16

    accuracy                           0.91        45
   macro avg       0.92      0.92      0.92        45
weighted avg       0.91      0.91      0.91        45
```

- Continuando com o estimador de Kernel pude comparar algumas outras configurações como a de h = 1 que obtive os seguintes valores

```
Accuracy: 0.8888888888888888
Precision: 0.9076923076923077
Recall: 0.8970588235294118
F1-Score:                precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.80      0.94      0.86        17
           2       0.92      0.75      0.83        16

    accuracy                           0.89        45
   macro avg       0.91      0.90      0.90        45
weighted avg       0.90      0.89      0.89        45
```

- Para o classificador K-NN configurei de forma ideal meu k como 11, onde obtive os valores abaixo

```
Accuracy: 0.9555555555555556
Precision: 0.9595588235294118
Recall: 0.9595588235294118
F1-Score:                precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.94      0.94      0.94        17
           2       0.94      0.94      0.94        16

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

- Nisso comparei meu K-NN com o valor de k igual a 5, onde a acurácia e precisão foram reduzida, como demonstrado na imagem a seguir

```
Accuracy: 0.9111111111111111
Precision: 0.9191176470588235
Recall: 0.9191176470588235
F1-Score:              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.88      0.88      0.88        17
           2       0.88      0.88      0.88        16

    accuracy                           0.91        45
   macro avg       0.92      0.92      0.92        45
weighted avg       0.91      0.91      0.91        45
```

- Observação: os cálculos referentes ao erro quadrático médio e o erro quadrado relativo estão disponíveis no final do código logo antes da matriz de confusão.

- Matriz de confusão do estimador de Kernel e K-NN, respectivamente