Introdução à Ciência dos Dados
Ciência da Computação – UFV/Florestal

**Aula Prática 09**
**Aprendizado Supervisionado**

**Objetivo:** utilizar a biblioteca scikit-learn para treinar e testar modelos de aprendizado supervisionado
**Pré-requisitos:** linguagem de programação Python, Linux, estatística
**Meta:** ao final da prática, o aluno será capaz de preparar dados e criar modelos de classificação

# Roteiro

Caso ainda não esteja instalado, instalar módulo Scikit-learn:

- Importar as bibliotecas a serem utilizadas

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

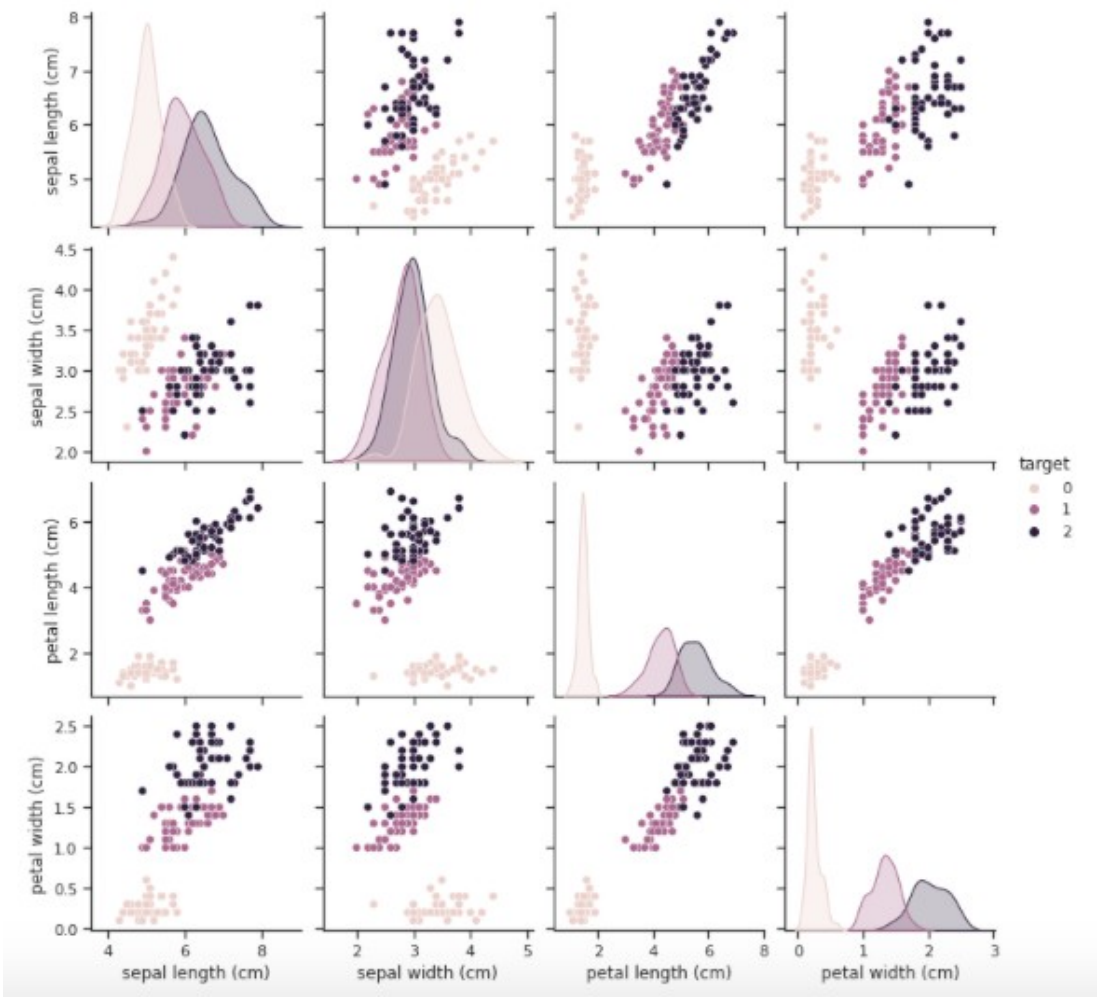
- O conjunto de dados está disponível diretamente na biblioteca scikit-learn

```
# The iris dataset is a classification task consisting in identifying 3 different types of
# irises (Setosa, Versicolour, and Virginica) from their petal and sepal length and width:
iris = datasets.load_iris()
iris_df = pd.DataFrame(np.c_[iris['data'], iris['target']],
                       columns= np.append(iris['feature_names'], ['target']))

# convert target to int
iris_df['target']=iris_df['target'].astype('int64')
```

- Vamos plotar os dados

```
# Plot the iris dataset
sns.set(style="ticks")
sns.pairplot(iris_df, hue="target")
plt.show()
```




- Vamos separar o conjunto de dados em treinamento e testes

```
# Split the dataset into two: train and test
X_train,X_test,Y_train,Y_test = train_test_split(
                iris_df[[u'sepal length (cm)', u'sepal width (cm)', u'petal length (cm)',u'petal width (cm)']],
                iris_df[u'target'],
                test_size=0.2, random_state=0)
```

- Aplicar o KNN para treinar um modelo e classificar os exemplos de testes

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
print("Accuracy = {0}%".format(100*np.sum(knn.predict(X_test) == Y_test)/len(Y_test)))
```

```
Accuracy = 96%
```

- Aplicar o SVM para treinar um modelo e classificar os exemplos de testes

```
# SVM
from sklearn import svm
svm = svm.SVC(kernel='linear', probability=True)
svm.fit(X_train, Y_train)
print("Accuracy = {0}%".format(100*np.sum(svm.predict(X_test) == Y_test)/len(Y_test)))
```

```
Accuracy = 100%
```

- Calcula a precisão e revogação

```
# Precision Recall
from sklearn.metrics import precision_recall_fscore_support

# KNN
print("KNN (precision, recall, fscore)")
precision_recall_fscore_support(Y_test, knn.predict(X_test), average=None)
```

```
KNN (precision, recall, fscore)
```

```
(array([ 1.        ,  1.        ,  0.85714286]),
 array([ 1.        ,  0.92307692,  1.        ]),
 array([ 1.        ,  0.96      ,  0.92307692]),
 array([11, 13,  6]))
```

```
# SVM
print("SVM (precision, recall, fscore)")
precision_recall_fscore_support(Y_test, svm.predict(X_test), average=None)
```

```
SVM (precision, recall, fscore)
```

```
(array([ 1.,  1.,  1.]),
 array([ 1.,  1.,  1.]),
 array([ 1.,  1.,  1.]),
 array([11, 13,  6]))
```

- Validação cruzada

```
# Cross-validation: KNN
from sklearn.model_selection import cross_val_score
scores = cross_val_score(knn,
                iris_df[[u'sepal length (cm)', u'sepal width (cm)', u'petal length (cm)',u'petal width (cm)']],
                iris_df[u'target'], cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.97 (+/- 0.05)
```

```
# Cross-validation: SVM
from sklearn.model_selection import cross_val_score
scores = cross_val_score(svm,
                iris_df[[u'sepal length (cm)', u'sepal width (cm)', u'petal length (cm)',u'petal width (cm)']],
                iris_df[u'target'], cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.98 (+/- 0.03)
```

# Atividades

Utilizar o conjunto de dados weatherHistory.csv, que contém o histórico de variáveis relacionadas ao clima da cidade de Szede, Hungria, entre 2006 e 2016.

1. Utilize técnicas de aprendizado de máquina supervisionado para criar modelos para prever o tipo de precipitação (*Precip Type*) de acordo com outras variáveis. Teste pelo menos três algoritmos e responda:

a) Qual a acurácia, precisão e revogação para cada rótulo (tipo de precipitação) para cada algoritmo testado? Existe algum algoritmo com resultados melhores?

b) É possível melhorar o algoritmo se forem selecionados alguns atributos em particular, ao invés de utilizar todos?

c) Você percebeu alguma diferença no tempo de processamento dos algoritmos? Qual foi o mais lento? E qual foi o mais rápido?

d) Mostre a matriz de confusão para os resultados (pesquise como fazer isso).