VALIDAÇÃO DE MODELOS E MÉTRICAS DE DESEMPENHO

_**DIVISÃO** DOS DADOS

```
> from sklearn.model_selection import train_test_split    # importa a função para divisão dos dados entre treino e teste
> x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, stratify = y)    # realiza a divisão dos dados entre treino e teste de forma estratificada
```


```
> x, x_teste, y, y_teste = train_test_split(x, y, stratify = y)
> x_treino, x_val, y_treino, y_val = train_test_split(x, y, stratify = y)    # realiza a divisão dos dados entre treino, validação e teste de forma estratificada
```

_**CRIANDO** UM MODELO

```
> from sklearn.tree import DecisionTreeClassifier      # importa o algoritmo de árvore de decisão
> modelo = DecisionTreeClassifier(max_depth = 10)      # inicializa um modelo de árvore de decisão com profundidade máxima de 10


> from sklearn.ensemble import RandomForestClassifier      # importa o algoritmo de floresta aleatória
> modelo = RandomForestClassifier(max_depth = 10)          # inicializa um modelo de floresta aleatória com profundidade máxima de 10
```

_**AJUSTANDO** UM MODELO

```
> modelo.fit(x_treino, y_treino)   # ajusta um modelo usando dados de treinamento
```

_**AVALIANDO** UM MODELO

```
> modelo.score(x_val, y_val) # avalia a taxa de acerto do modelo usando dados de validação
```

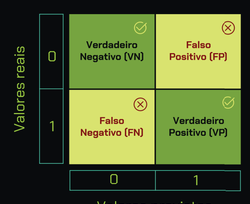
_**FAZENDO** PREVISÕES

```
> y_previsto = modelo.predict(x_val)      # fazendo previsões do modelo a partir dos dados de validação
```

_**MATRIZ** DE CONFUSÃO

```
> from sklearn.metrics import confusion_matrix     # importa a função para criação da matriz de confusão
> from sklearn.metrics import ConfusionMatrixDisplay  # importa a função para criação da visualização da matriz de confusão


> matriz_confusao = confusion_matrix(y_val, y_previsto)    # gerando uma matriz de confusão
> print(matriz_confusao)
> ConfusionMatrixDisplay(confusion_matrix = matriz_confusao)  # gerando a visualização a partir de uma matriz de confusão
> ConfusionMatrixDisplay.from_predictions(y_val, y_previsto)  # gerando a visualização da matriz de confusão a partir dos dados reais e previstos
```



_**MÉTRICAS** DE DESEMPENHO

```
> from sklearn.metrics import accuracy_score, precision_score,
  recall_score, f1_score, roc_auc_score, average_precision_score      # importa as funções para calcular as métricas de acurácia, precisão, recall, f1-score, AUC e AP
> from sklearn.metrics import classification_report      # importa a função para gerar o relatório de métricas de classificação


> accuracy_score(y_val, y_previsto)    # calcula a acurácia do modelo usando os dados reais e previstos
```

$$\text{Acurácia} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}$$

```
> precision_score(y_val, y_previsto)   # calcula a precisão do modelo usando os dados reais e previstos
```

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}}$$

```
> recall_score(y_val, y_previsto)    # calcula o recall do modelo usando os dados reais e previstos
```

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

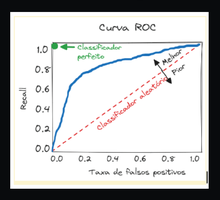
```
> f1_score(y_val, y_previsto) # calcula o f1-score do modelo usando os dados reais e previstos
```

$$\text{F1} = 2 \cdot \frac{\text{precisão} \cdot \text{recall}}{\text{precisão} + \text{recall}}$$

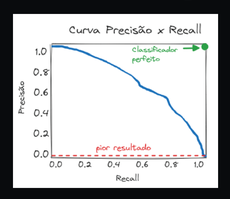
```
> roc_auc_score(y_val, y_previsto)  # calcula o AUC do modelo usando os dados reais e previstos
> average_precision_score(y_val, y_previsto)  # calcula a precisão média (AP) do modelo usando os dados reais e previstos
> classification_report(y_val, y_previsto)   # gera o relatório de métricas de classificação usando os dados reais e previstos
```

_**GRÁFICOS** DE AVALIAÇÃO DE DESEMPENHO

```
> from sklearn.metrics import RocCurveDisplay      # importa a função para gerar a curva ROC
> RocCurveDisplay.from_predictions(y_val, y_previsto);  # gera o gráfico da curva ROC usando os dados reais e previstos
```




```
> from sklearn.metrics import PrecisionRecallDisplay  # importa a função para gerar a curva precisão x recall
> PrecisionRecallDisplay.from_predictions(y_val, y_previsto);  # gera o gráfico da curva precisão x recall usando os dados reais e previstos
```



_**VALIDAÇÃO** CRUZADA

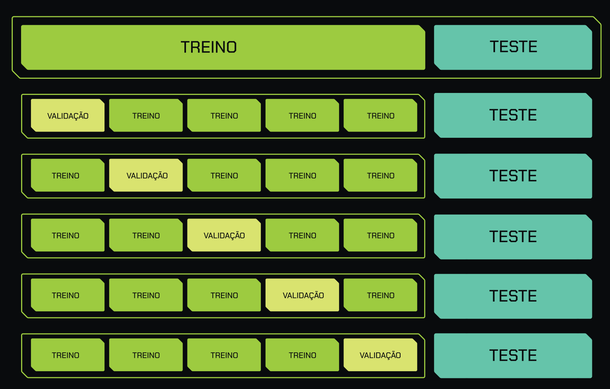
```
> from sklearn.model_selection import KFold     # importa a função do KFold
> kf = KFold(n_splits = 5, shuffle = True)   # inicializa o KFold com 5 folds e embaralhamento dos dados


> from sklearn.model_selection import StratifiedKFold   # importa a função do StratifiedKFold
> skf = StratifiedKFold(n_splits = 5, shuffle = True)  # inicializa o KFold estratificado com 5 folds e embaralhamento dos dados


> from sklearn.model_selection import LeaveOneOut    # importa a função do LeaveOneOut
> loo = LeaveOneOut()  # inicializa o LeaveOneOut


> from sklearn.model_selection import cross_validate # importa a função de validação cruzada para resultados das métricas e tempo de execução
> cross_validate(modelo, x, y, cv = kf)  # realiza a validação cruzada para resultados das métricas e tempo de execução


> from sklearn.model_selection import cross_val_score   # importa a função de validação cruzada para resultados das métricas
> cross_val_score(modelo, x, y, cv = kf)  # realiza a validação cruzada para resultados das métricas
```



_**BALANCEAMENTO** DE DADOS

```
> from imblearn.over_sampling import SMOTE      # importa a função de oversampling com SMOTE
> oversample = SMOTE()
> x_balanceado, y_balanceado = oversample.fit_resample(x, y)    # inicializa e faz o oversampling de dados com SMOTE


> from imblearn.under_sampling import NearMiss    # importa a função de undersampling com NearMiss

> undersample = NearMiss()
> x_balanceado, y_balanceado = undersample.fit_resample(x, y)  # inicializa e faz o undersampling de dados com NearMiss
```

_**PIPELINE** DE DADOS

```
> from imblearn.pipeline import Pipeline as imbpipeline  # importa a função de criação de pipelines
> pipeline = imbpipeline([('oversample', SMOTE()), ('modelo', modelo)])  # cria um pipeline de oversampling com SMOTE e um modelo de ML
> pipeline = imbpipeline([('undersample', NearMiss(version = 3)), ('arvore', modelo)])  # cria um pipeline de undersampling com NearMiss e um modelo de ML
```

*clique nos ícones para **acessar** as respectivas **documentações***

alura | Escola de DATA SCIENCE