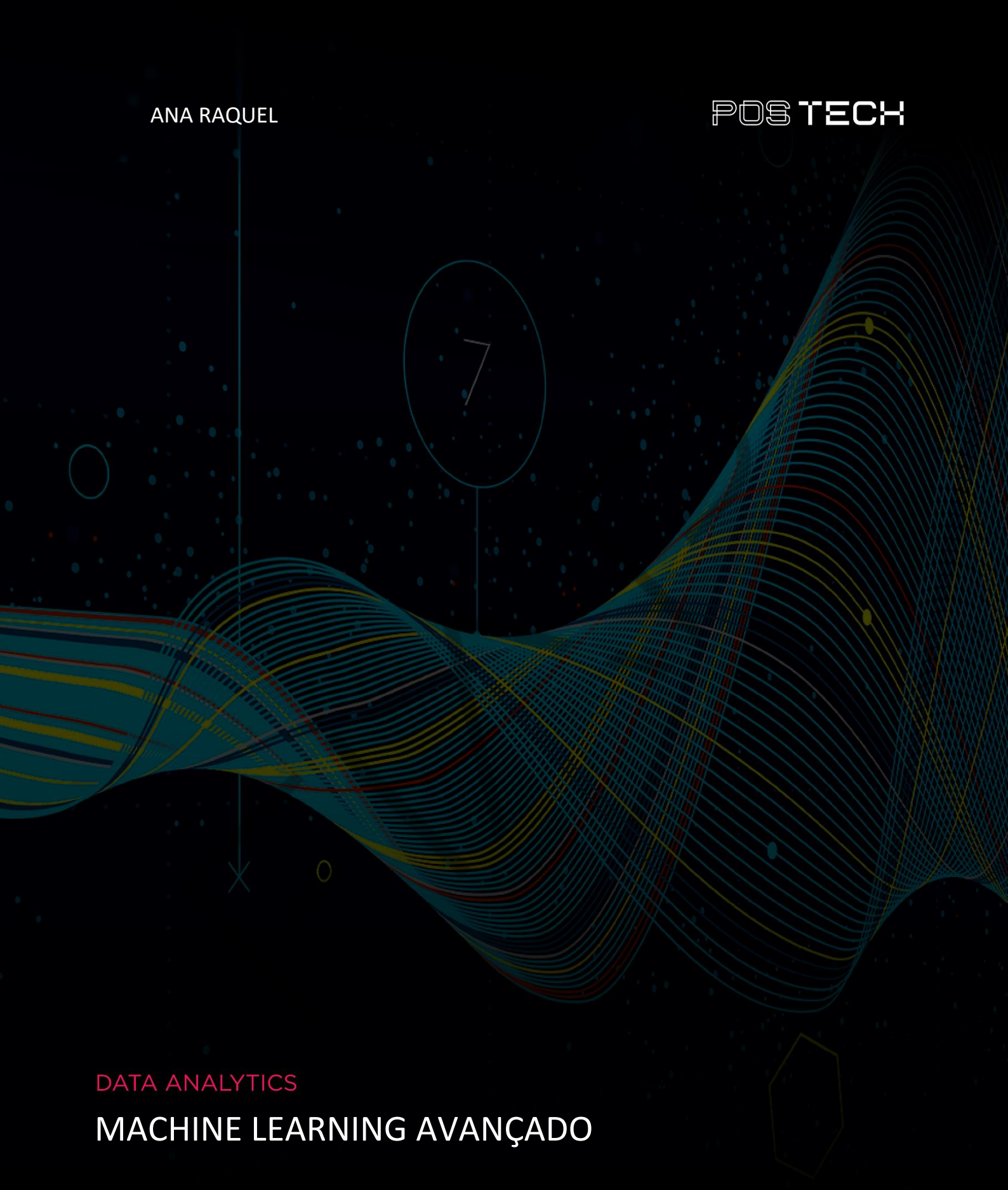
ANA RAQUEL



DATA ANALYTICS

MACHINE LEARNING AVANÇADO

AULA 05

SUMÁRIO

O QUE VEM POR AÍ?

Você aprendeu a importância de separar as bases de dados para modelos de classificação e regressão em treino e teste, mas como garantir que seu modelo funcione com dados nunca visto antes? Será que existe uma outra técnica que valide com mais precisão? Nesta aula, você aprenderá sobre a técnica de “Validação Cruzada”.

HANDS ON

Veremos como aplicar a técnica de validação cruzada para selecionar os melhores modelos e hiperparâmetros para resolver problemas de classificação. Vamos lá?

Para essa aula, temos um notebook para você. Acesse abaixo:

- Notebook 1

SAIBA MAIS

A validação cruzada, também conhecida como **K-Fold**, divide de forma aleatória os dados de um conjunto de treinamento em subconjuntos distintos (os chamados folds), e então treina e avalia o modelo em cada subconjunto de dados, obtendo resultados diferentes de acordo com a iteração de K escolhida.

k=1	k=2	k=3	k=4
Test	Train	Train	Train
Train	Test	Train	Train
Train	Train	Test	Train
Train	Train	Train	Test

Figura 1 - Exemplo K-Fold
Fonte: Elaborado pela autora (2023)

A validação cruzada nos retorna um score de cada fold treinado. Essa técnica compara a performance de vários modelos, dando a possibilidade de escolhermos qual modelo foi mais preciso. Além de trazer ganho em análise de performance de modelos, a validação cruzada também pode ser utilizada para selecionar os melhores hiperparâmetros de um modelo.

Vamos analisar como esse modelo funcionaria no Python:

```
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import KFold

kfold  = KFold(n_splits=5, shuffle=True)

result = cross_val_score(modelo_classificador, x, y, cv = kfold)

print("K-Fold (R^2) Scores: {0}".format(result))

print("Mean R^2 for Cross-Validation K-Fold:
{0}".format(result.mean()))
```

```
[46] from sklearn.model_selection import cross_val_score
     from sklearn.model_selection import KFold

[47] kfold  = KFold(n_splits=5, shuffle=True) # shuffle=True, Shuffle (embaralhar) the data.
     result = cross_val_score(modelo_classificador, x, y, cv = kfold)

     print("K-Fold (R^2) Scores: {0}".format(result))
     print("Mean R^2 for Cross-Validation K-Fold: {0}".format(result.mean()))

     K-Fold (R^2) Scores: [0.74193548 0.79032258 0.88709677 0.80645161 0.81967213]
     Mean R^2 for Cross-Validation K-Fold: 0.8090957165520887
```

Figura 2 - Aplicação da validação cruzada no algoritmo KNN
Fonte: Elaborado pela autora (2023)

Basicamente, em **K-Fold,** escolhemos em **n_splits** o número total de subconjuntos de dados que serão divididos, e com a opção **shuffle** conseguimos embaralhar os dados. Como próximo passo, passamos o modelo que queremos testar com a base nos dados separados entre variáveis características em x e variável preditora em y. Com a técnica de validação cruzada, você pode comparar os scores de vários modelos de algoritmos diferentes e avaliar qual obtém a melhor performance.

Vamos analisar como poderíamos aplicar essa técnica com vários algoritmos ao mesmo tempo:

```
def AplicaValidacaoCruzada(x_axis, y_axis):
  # Linear Models.
  from sklearn.neighbors import KNeighborsClassifier  # k-vizinhos mais
próximos (KNN)
  from sklearn.ensemble import RandomForestClassifier # RandomForest
  from sklearn.svm import SVC                         # Maquina de
Vetor Suporte SVM


  # Cross-Validation models.
  from sklearn.model_selection import cross_val_score
  from sklearn.model_selection import KFold


  # Configuração de KFold.
  kfold  = KFold(n_splits=10, shuffle=True)
```

```
  # Axis
  x = x_axis
  y = y_axis

  # Criando os modelos

  # KNN
  knn = KNeighborsClassifier(n_neighbors=9, metric= 'cosine',
weights='distance')
  knn.fit(x_train_scaled, y_train)

  # SVM
  svm = SVC()
  svm.fit(x_train_scaled, y_train)

  # RandomForest
  rf = RandomForestClassifier(random_state=7)
  rf.fit(x_train_scaled, y_train)

  # Applyes KFold to models.
  knn_result = cross_val_score(knn, x, y, cv = kfold)
  svm_result = cross_val_score(svm, x, y, cv = kfold)
  rf_result = cross_val_score(rf, x, y, cv = kfold)

  # Creates a dictionary to store Linear Models.
  dic_models = {
    "KNN": knn_result.mean(),
    "SVM": svm_result.mean(),
    "RF": rf_result.mean()
  }
  # Select the best model.
  melhorModelo = max(dic_models, key=dic_models.get)
```

```
  print("KNN (R^2): {0}\nSVM (R^2): {1}\nRandom Forest (R^2):
{2}".format(knn_result.mean(), svm_result.mean(), rf_result.mean()))

  print("O melhor modelo é : {0} com o valor: {1}".format(melhorModelo,
dic_models[melhorModelo]))
```

```
  AplicaValidacaoCruzada(x, y)

KNN (R^2): 0.7795698924731183
SVM (R^2): 0.85752688172043
Random Forest (R^2): 0.8380645161290323
O melhor modelo é : SVM com o valor: 0.85752688172043
```

Figura 3 - Buscando os melhores algoritmos
Fonte: Elaborado pela autora (2023)

Perceba que podemos validar de forma bem rápida a validação cruzada de vários algoritmos ao mesmo tempo e encontrar qual algoritmo nos traz a melhor performance.

Com a técnica **Gridsearch** também conseguimos utilizar a validação cruzada para escolher os melhores hiperparâmetros. Essa técnica consiste em treinar vários hiperparâmetros configurados em **param_grid**, como se fosse uma busca dos melhores hiperparâmetros utilizando a força bruta:

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, accuracy_score, f1_score


# Parâmetros testados
param_grid = {'n_neighbors':[9,14],
              'weights': ['uniform', 'distance'],
'metric':['cosine', 'euclidean', 'manhattan'] }


gs_metric = make_scorer(accuracy_score, greater_is_better=True)


grid = GridSearchCV(KNeighborsClassifier(),
                    param_grid=param_grid,
```

```
                    scoring=gs_metric,
                    cv=5, n_jobs=4, verbose=3)


grid.fit(x_train_scaled, y_train)
knn_params = grid.best_params_
print('KNN', knn_params)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
KNN {'metric': 'cosine', 'n_neighbors': 9, 'weights': 'distance'}
```

Figura 4 - Buscando os melhores hiperparâmetros em um algoritmo
Fonte: Elaborado pela autora (2023)

O QUE VOCÊ VIU NESTA AULA?

Validação cruzada, K-Fold e Gridsearch.

Daqui em diante, é importante que você replique os conhecimentos adquiridos nesta aula para fortalecer ainda mais as suas bases e conhecimentos.

IMPORTANTE: não esqueça de praticar com o desafio da disciplina, para que assim você possa aprimorar os seus conhecimentos!

Você não está só nesta jornada! Te esperamos no Discord e nas lives com os nossos especialistas, onde você poderá tirar dúvidas, compartilhar conhecimentos e estabelecer conexões!

REFERÊNCIAS

DOCUMENTAÇÃO SCIKIT-LEARN. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 11 abr 2023.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 2nd Edition. [s.l.]: O'Reilly Media, Inc., 2019.

PALAVRAS-CHAVE

VALIDAÇÃO CRUZADA. K-FOLD. GRIDSEARCH.