Validação Holdout na prática com Python

Ciclo	Ciclo 04: As garantias de aprendizado
# Aula	29
Created	@February 24, 2023 8:49 AM
Done	☑
Ready	☑

Objetivo da Aula:

- [ ] Validação Holdout na prática
- [ ] Resumo
- [ ] Próxima aula

Conteúdo:

▼ 1. Validação Holdout na prática

```
# Import libraries
import numpy                          as np
from sklearn    import datasets       as ds
from sklearn    import tree           as tr
from sklearn    import metrics        as mt
from sklearn    import model_selection as ms
from matplotlib import pyplot         as plt

# 1.0 Treinamento como o Joaquim: O DS Novato

## Dados sintéticos para produção
n_samples = 20000
n_features = 2
n_informative = 2
n_redundant = 0
random_state = 0

# Dados para treinamento
X, y = ds.make_classification( n_samples=n_samples, n_features=n_features,
                               n_informative=n_informative, n_redundant=n_redundant,
                               random_state=random_state )

# Dados para produção
X, X_prod, y, y_prod = ms.train_test_split( X, y, test_size=0.2 )


## Não há separação dos Dados
# Modelo treinado e validado com o dataset de Treinamento
model = tr.DecisionTreeClassifier( max_depth=38 )
model.fit( X, y )

# Previsão sobre os dados de treinamento
yhat = model.predict( X )
acc = mt.accuracy_score( y, yhat )
print( "Accuracy Over Training: {}".format( acc ) )
```

```
## Publicação do Modelo em Produção
# Previsão sobre os dados de treinamento
yhat_prod = model.predict( X_prod )
acc_prod = mt.accuracy_score( y_prod, yhat_prod )
print( "Accuracy Over Production: {}".format( acc_prod ) )


# 2.0 Estratégia Treino-Teste
## Separação entre Treino e Teste
X_train, X_test, y_train, y_test = ms.train_test_split( X, y, test_size=0.2, random_state=random_state )

## Modelo treinado e validado com o dataset de Treinamento
model = tr.DecisionTreeClassifier( max_depth=58 )
model.fit( X_train, y_train )

## Previsão sobre os dados de treinamento
yhat_test = model.predict( X_test )
acc_test = mt.accuracy_score( y_test, yhat_test )
print( "Accuracy Over Test: {}".format( acc_test ) )


# Escolha de parâmetros do algoritmo
## Modelo treinado e validado com o dataset de Treinamento
values = [i for i in range( 1, 60 )]
test_scores = list()

for i in values:
    model = tr.DecisionTreeClassifier( max_depth=i )
    model.fit( X_train, y_train )

    # Previsão sobre os dados de test
    yhat_test = model.predict( X_test )
    acc_test = mt.accuracy_score( y_test, yhat_test )

    test_scores.append( acc_test )


## plot of train and test scores vs tree depth
plt.plot( values, test_scores, '-o', label='Test' )
plt.legend()
plt.show()


# Publicação do Modelo em Produção
## Modelo treinado e validado com o dataset de Treinamento
model_last = tr.DecisionTreeClassifier( max_depth=7 )
model_last.fit( np.concatenate( (X_train, X_test) ), np.concatenate((y_train, y_test)) )
model_last.fit( X_train, y_train )


## Previsão sobre os dados de produção
yhat_prod = model_last.predict( X_prod )
acc_prod = mt.accuracy_score( y_prod, yhat_prod )
print( "Accuracy Over Production: {}".format( acc_prod ) )


# 3.0 Estratégia Treino-Validation-Teste
## Separação entre Treino e Teste
X_train, X_val, y_train, y_val = ms.train_test_split( X_train, y_train, test_size=0.2, random_state=random_state )

## Modelo treinado e validado com o dataset de Treinamento
values = [i for i in range( 1, 60 )]
val_scores = list()

for i in values:
    model = tr.DecisionTreeClassifier( max_depth=i )
    model.fit( X_train, y_train )

    # Previsão sobre os dados de treinamento
    yhat_val = model.predict( X_val )
    acc_val = mt.accuracy_score( y_val, yhat_val )

    val_scores.append( acc_val )

## plot of train and test scores vs tree depth
```

```
plt.plot( values, val_scores, '-o', label='Validation' )
plt.legend()
plt.show()

## Previsão sobre os dados de validacao
yhat_val = model.predict( X_val )
acc_val = mt.accuracy_score( y_val, yhat_val )
print( "Accuracy Over Validation: {}".format( acc_val ) )

## Modelo treinado e validado com o dataset de Treinamento
model_last = tr.DecisionTreeClassifier( max_depth=7 )
model_last.fit( np.concatenate( (X_train, X_val) ), np.concatenate((y_train, y_val)) )

## Previsão sobre os dados de test
yhat_test = model_last.predict( X_test )
acc_test = mt.accuracy_score( y_test, yhat_test )
print( "Accuracy Over Test: {}".format( acc_test ) )

# Previsão sobre os dados de treinamento
yhat_prod = model_last.predict( X_prod )
acc_prod = mt.accuracy_score( y_prod, yhat_prod )
print( "Accuracy Over Production: {}".format( acc_prod ) )
```

▼ 2. Resumo

1. A performance do algoritmo nos dados de teste deve ser próxima da performance do algoritmo nos dados de produção.
2. A validação Holdout deve ser sempre utilizada, a menos que o conjunto de dados seja muito pequeno.

▼ 3. Próxima aula

O problema do Overfitting na Classificição