# O problema do Overfitting na Classificação

| | |
|---|---|
| Ciclo | Ciclo 04: As garantias de aprendizado |
| # Aula | 30 |
| Created | @February 16, 2023 2:25 PM |
| Done | ☑ |
| Ready | ☑ |

## Objetivo da Aula:

- [ ] Parte I - A viagem ao país desconhecido
- [ ] O problema da generalização
- [ ] Classificação com Overfitting
- [ ] Regressão com Overfitting
- [ ] Resumo
- [ ] Próxima aula

## Conteúdo:

## ▼ 1. Parte I - A viagem ao país desconhecido

Joaquim sonhava em viajar para um lugar desconhecido, longe de sua casa. Então, um dia, ele decidiu embarcar em uma aventura e partiu para um país que nunca tinha visitado.

Assim que chegou no aeroporto, Joaquim pegou o primeiro táxi que estava estacionado próximo ao local de desembarque dos vôo, para que o levasse até o hotel em que tinha feito reservas.

A viagem de taxi foi rápida, mas o valor da corrida ficou completamente desproporcional e Joaquim teve que pagar o valor pedido pelo motorista. Joaquim chegou no seu hotel, fez o check-in e foi para a sua acomodação cheio de raiva e frustração. Ele ligou para sua família e amigos, xingou muito no Twitter, fez um Stories no Instagram dizendo como todos os taxistas daquele país eram ladrões.

## ▼ 2. O problema da generalização

Generalização é algo que o ser humano faz muito frequentemente, ele usa um único exemplo ou poucos exemplos e define o comportamento geral de um seguimento. “Todos os taxistas são ladrões” foi uma afirmação baseada em um único exemplo.

Infelizmente, os algoritmos também podem cair nessa armadilha se não tomarmos cuidados. Em Machine Learning, esse problemas é chamado de “overfitting”, do português, sobreajuste.

Overfitting significa que o modelo tem uma boa performance nos dados de treinamento e validação, mas não consegue manter a mesma performance para dados não vistos, ou seja, não consegue generalizar. Ele comete muitos erros para exemplos não vistos.

## ▼ 3. Classificação com Overfitting

```
# 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 )

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-Validacao-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_st
ate=random_state )

# Separação entre Treino e Validacao
X_train, X_val, y_train, y_val = ms.train_test_split( X_train, y_train, test_size=0.2,
random_state=random_state )
```

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

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

    # Previsão sobre os dados de treinamento
    yhat_train = model.predict( X_train )
    acc_train = mt.accuracy_score( y_train, yhat_train )

    train_scores.append( acc_train )

    # Previsão sobre os dados de test
    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, train_scores, '-o', label='Train' )
plt.plot( values, val_scores,  '-o', label='Validacao' )
plt.legend()
plt.show()

# Modelo treinado e validado com o dataset de Treinamento
model_last = tr.DecisionTreeClassifier( max_depth=4 )
model_last.fit( X_train , y_train )

# Previsão sobre os dados de treinamento
yhat_train = model_last.predict( X_train )
acc_train = mt.accuracy_score( y_train, yhat_train )
print( "Accuracy Over Training: {}".format( acc_train ) )

# Previsão sobre os dados de validacao
yhat_val = model_last.predict( X_val )
acc_val = mt.accuracy_score( y_val, yhat_val )
print( "Accuracy Over Validacao: {}".format( acc_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 Testing: {}".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 Producao: {}".format( acc_prod ) )
```

## ▼ 4. Resumo

1. O aprendizado supervisionada do tipo Regressão é utilizado para duas finalidades: Estudar o fenômeno ou criar um modelo para fazer previsão.
2. Tarefas de Regressão usam uma variável alvo ou resposta como representação do fenômeno.

## ▼ 5. Próxima aula

O problema do Overfitting na Regressão