Curso de Ciência de Dados Aula4

Para receber por e-mail o(s) arquivo(s) utilizados na aula, preencha:

Seu melhor e-mail

Enviar

Nessa aula nós vamos falar sobre aprendizado de máquina para classificação, então vamos ver alguns modelos de classificação para prever o valor de uma categoria.

Nesse exemplo para o nosso modelo de aprendizado de máquina nós vamos utilizar novamente a nossa base do titanic.

Aprendizagem Supervisionada

Para essa aula nós vamos utilizar a aprendizagem supervisionada, que é quando nós temos tanto os valores de entrada quanto os valores de saída (valor target).

Essa primeira parte de explicar o que são cada uma das informações e o que temos em cada uma das colunas nós já vimos na **aula anterior**, então essa parte você pode dar uma olhada lá!

Podemos começar o tratamento retirando a coluna "Cabin" que tem muitos valores vazios

```
# Eliminando a coluna 'Cabin'
base = base.drop('Cabin',axis=1)
```

```
base.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

Removendo a coluna “Cabin” da base de dados

E então retirar todos os valores vazios

- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html

```
# Eliminando valores vazios
base = base.dropna()
```

```
base.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  712 non-null    int64
 1   Survived     712 non-null    int64
 2   Pclass       712 non-null    int64
 3   Name         712 non-null    object
 4   Sex          712 non-null    object
 5   Age          712 non-null    float64
 6   SibSp        712 non-null    int64
 7   Parch        712 non-null    int64
 8   Ticket       712 non-null    object
 9   Fare         712 non-null    float64
 10  Embarked     712 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 66.8+ KB
```

Eliminando os valores vazios

Como nós vamos simplificar esse tratamento nesse exemplo nós vamos eliminar também todas as colunas com o tipo “object”, pois vamos deixar apenas as colunas com o tipo inteiro e ponto flutuante.

E também retirar as colunas que não são valores inteiros

```
# Selecionando as colunas com o tipo "object"
colunas = base.dtypes[base.dtypes.values == 'object'].index
```

```
# Eliminando essas colunas
base = base.drop(colunas,axis=1)
```

```
# Verificando novamente as informações da base
base.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  712 non-null    int64
 1   Survived     712 non-null    int64
 2   Pclass       712 non-null    int64
 3   Age          712 non-null    float64
 4   SibSp        712 non-null    int64
 5   Parch        712 non-null    int64
 6   Fare         712 non-null    float64
dtypes: float64(2), int64(5)
memory usage: 44.5 KB
```

Removendo as colunas do tipo objeto

Com isso nossa base já está quase pronta para utilizarmos. O próximo passo é onde vamos começar a utilizar alguns dos modelos de classificação.

Mas para isso vamos precisar da biblioteca scikit-learn, que é uma biblioteca para análise preditiva de dados. Você pode acessar o site dela no link abaixo:

https://scikit-learn.org/stable/

- Construído em NumPy, SciPy e Matplotlib
- Disponível em: https://scikit-learn.org/

```
# Definindo o X e y para o treino
X = base.drop('Survived',axis=1)
y = base.Survived
```

Definindo x e y de treino

Antes de fazer qualquer uso de modelos de dados nós vamos definir o x e y para o treino, onde o x vai ser a nossa base ser a coluna “Survived” que é o que queremos descobrir.

E o y será a coluna survived para que possamos fazer uma comparação com os modelos de classificação.

Para esse exemplo nós vamos utilizar 3 modelos de classificação, o KNN, o modelo árvore de decisão e a regressão logística.

Utilizando o KNN

- https://scikit-learn.org/stable/modules/generated/sklearn.neigh

```
# Importando o KNN
from sklearn.neighbors import KNeighborsClassifier
```

```
# Criando o nosso classificador
neigh = KNeighborsClassifier(n_neighbors=3)
```

```
# Fazendo o fit com os dados
neigh.fit(X, y)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
# Avaliando o modelo
neigh.score(X, y)
```

```
0.7907303370786517
```

Usando o modelo KNN

OBS: Não se preocupe, pois todos os links e códigos estão disponíveis para download, então se precisar acessar a documentação de cada biblioteca no arquivo estão todos os links e está tudo separado para que você consiga acompanhar!

Utilizando a Árvore de Decisão

- https://scikit-learn.org/stable/modules/tree.html#classification

```
# Importando a árvore de decisão
from sklearn import tree
```

```
# Criando o nosso classificador
clfArvore = tree.DecisionTreeClassifier(random_state=0)
```

```
# Fazendo o fit com os dados
clfArvore = clfArvore.fit(X, y)
```

```
# Avaliando o modelo
clfArvore.score(X,y)
```

```
1.0
```

Usando o modelo Árvore de Decisão

Por fim vamos utilizar a regressão logística. Pode ficar calmo que ainda vamos fazer as comparações para que esses valores façam sentido.

```
# Importando a regressão logística
from sklearn.linear_model import LogisticRegression
```

```
# Criando o nosso classificador e fazendo o fit com os dados
clfLog = LogisticRegression(random_state=0,max_iter=1000).fit(X, y)
```

```
# Avaliando o modelo
clfLog.score(X,y)
```

```
0.7064606741573034
```

Usando a Regressão Logística

Feito isso nós vamos fazer uma avaliação dos modelos de classificação, então agora nós vamos fazer o mesmo tratamento que fizemos anteriormente, só que agora utilizando a base de teste.

Avaliando modelos de classificação

- https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
- Para isso, vamos usar os dados de teste
  - Base: `titanic_test.csv`

```
# Importando e visualizando a base de teste
teste = pd.read_csv('titanic_test.csv')
```

```
# Fazendo os mesmos tratamentos
teste = teste.drop('Cabin',axis=1)
teste = teste.dropna()
teste = teste.drop(colunas,axis=1)
```

```
# Verificando as informações da base
teste.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 331 entries, 0 to 415
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  331 non-null    int64
 1   Survived     331 non-null    int64
 2   Pclass       331 non-null    int64
 3   Age          331 non-null    float64
 4   SibSp        331 non-null    int64
 5   Parch        331 non-null    int64
 6   Fare         331 non-null    float64
dtypes: float64(2), int64(5)
memory usage: 20.7 KB
```

Avaliando os modelos de classificação

Aqui fizemos o mesmo tratamento que foi feito anteriormente, até porque precisamos do mesmo tratamento para que os resultados sejam compatíveis.

```
# Separando X e y da base
X_teste = teste.drop('Survived',axis=1)
y_teste = teste.Survived
```

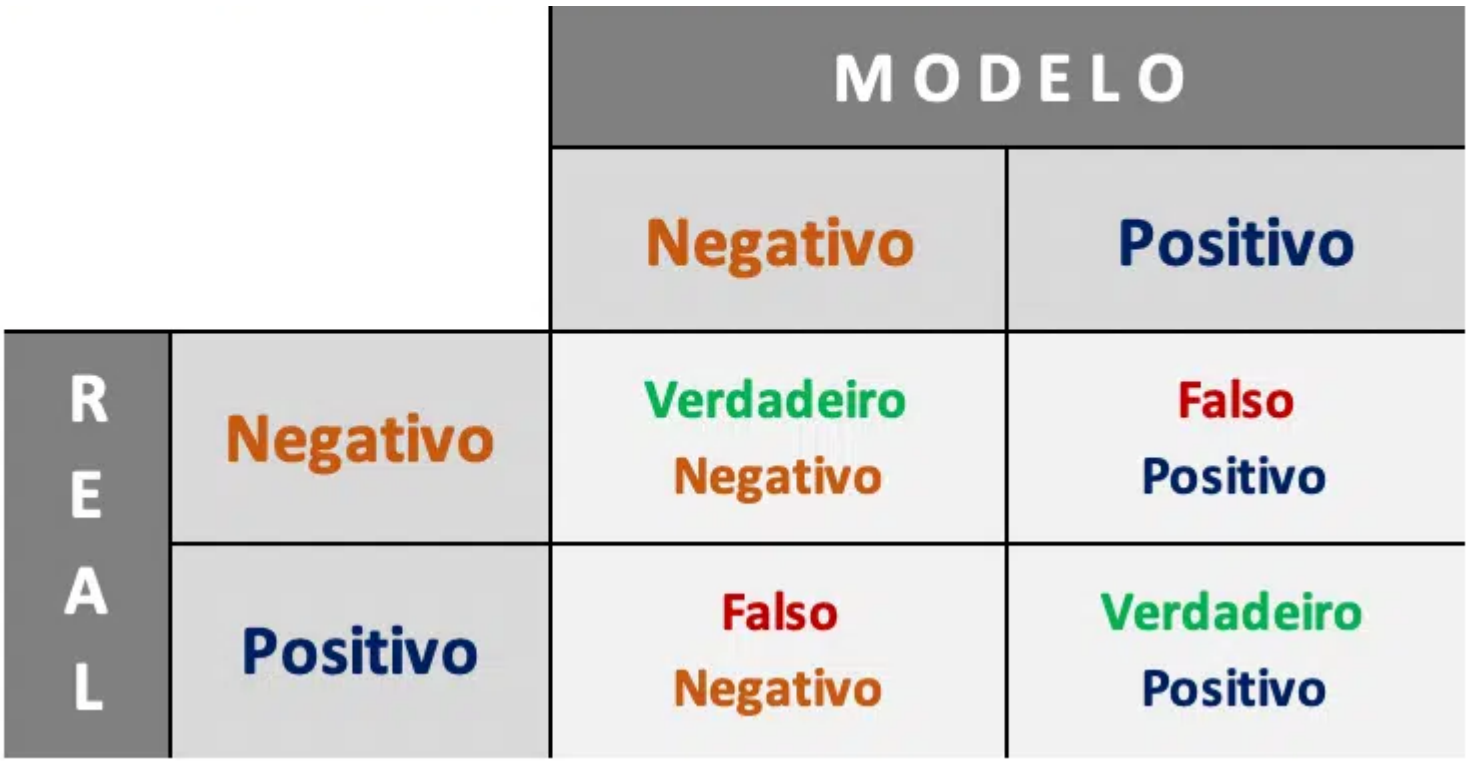
```
# Fazendo a predição com o KNN
pred_KNN = neigh.predict(X_teste)
```

```
# Predict com a Árvore de Decisão
pred_Arvore = clfArvore.predict(X_teste)
```

```
# Predict com Regressão Logística
pred_Log = clfLog.predict(X_teste)
```

Avaliando os modelos com a base de teste

Feito isso fizemos a análise com os 3 modelos de classificação. Agora nós vamos utilizar a **matriz de confusão** para fazer uma análise desses dados para comparar o real com o modelo.



Visual da matriz de confusão

```
# Importando a matriz de confusão
from sklearn.metrics import confusion_matrix
```

```
# Verificando a matriz para o KNN
confusion_matrix(y_teste, pred_KNN)
```

```
array([[ 68, 136],
       [ 35,  92]])
```

0 1

0 68 136

1 35 92

```
# Para a Árvore de Decisão
confusion_matrix(y_teste, pred_Arvore)
```

```
array([[110,  94],
       [ 43,  84]])
```

```
# E para a Regressão Logística
confusion_matrix(y_teste, pred_Log)
```

```
array([[131,  73],
       [ 64,  63]])
```

Cálculo da matriz de confusão para cada um dos modelos

Aqui já temos uma visualização de que o KNN prevê melhor as pessoas que vão sobreviver, enquanto a regressão logística prevê melhor as pessoas que não vão sobreviver.

Além de avaliar essas informações pela matriz de confusão nós podemos utilizar a **acurácia**, a **precisão** o **recall** para definir qual modelo é melhor para esse caso.

Na acurácia nós vamos verificar quantos valores nós acertamos independente se acertamos para o positivo ou negativo. Isso quer dizer que é um cálculo em relação ao total.

```
# Importando a métrica de acurácia
from sklearn.metrics import accuracy_score
```

```
# Verificando o erro para o KNN
accuracy_score(y_teste, pred_KNN)
```

```
0.48338368580060426
```

```
# Para a Árvore de Decisão
accuracy_score(y_teste, pred_Arvore)
```

```
0.5861027190332326
```

```
# E para a Regressão Logística
accuracy_score(y_teste, pred_Log)
```

```
0.5861027190332326
```

Acurácia dos modelos

Aqui você já nota que os dois últimos modelos acertaram uma mesma quantidade, enquanto o primeiro modelo já teve uma quantidade de acertos menor.

Agora na precisão nós vamos verificar dos valores positivos (falso-positivo e verdadeiro-positivo) quantos nós acertamos.

Precisão

- https://scikit-learn.org/stable/modules/generated/sklea

```
# Importando o precision_score
from sklearn.metrics import precision_score
```

```
# Verificando o erro para o KNN
precision_score(y_teste, pred_KNN)
```

```
0.40350877192982454
```

```
# Para a Árvore de Decisão
precision_score(y_teste, pred_Arvore)
```

```
0.47191011235955055
```

```
# E para a Regressão Logística
precision_score(y_teste, pred_Log)
```

```
0.4632352941176471
```

Precisão dos modelos

Aqui o modelo de árvore de decisão teve uma maior precisão, então com esses parâmetros até o momento ele seria o melhor modelo dentre os que testamos.

O recall vai fazer uma relação entre o verdadeiro (positivo) e o falso (negativo). Um bom exemplo é quando estamos tratando de fraude, então queremos encontrar a maior quantidade de fraudes.

```
# Importando o recall_score
from sklearn.metrics import recall_score
```

```
# Verificando o erro para o KNN
recall_score(y_teste, pred_KNN)
```

```
0.7244094488188977
```

```
# Para a Árvore de Decisão
recall_score(y_teste, pred_Arvore)
```

```
0.6614173228346457
```

```
# E para a Regressão Logística
recall_score(y_teste, pred_Log)
```

```
0.49606299212598426
```

Recall dos modelos

E por mais que a precisão seja muito boa, se o recall for muito baixo nós não vamos conseguir encontrar essas fraudes.

Então no nosso modelo essas seriam pessoas que sobreviveram, mas o nosso modelo classificou que essas pessoas não sobreviveram.

OBS: Então você já nota que não basta apenas ter todos os valores mais altos que já um ótimo modelo, isso depende muito da análise que nós estamos fazendo. Isso que dizer que para cada caso você vai ter que analisar quais são as melhores métricas a serem analisadas.

Nesse caso se quiséssemos identificar as pessoas que de fato sobreviveram e o modelo definiu como não sobrevivente poderíamos utilizar o recall, que para o KNN tem o valor mais alto.

Então esse modelo já conseguiria fazer essa verificação com maior facilidade. Assim você consegue começar a entender que não existe um modelo certo ou um modelo perfeito, isso tudo depende da sua análise e do que você precisa.

Aula 5 – Importância do Tratamento de Dados

Caso prefira esse conteúdo no formato de vídeo-aula, assista ao vídeo abaixo ou acesse o nosso canal do YouTube!

Curso de Ciência de Dados Aula5

Para receber por e-mail o(s) arquivo(s) utilizados na aula, preencha: