Sabia que é possível melhorar a acurácia de um modelo fazendo o tratamento de dados?

Na última aula nós vimos como avaliar os modelos de classificação, e agora vamos ver como fazer para melhorar os parâmetros com o tratamento de dados!

Tratamento de Dados em Python

Antes de iniciar vou colocar aqui o que fizemos na última aula tanto para a nossa base de treino quanto para a base de teste.

Etapa 0: Relembrando o que fizemos na última aula

```
# Para a base de treino
base = pd.read_csv('titanic_train.csv')
base = base.drop('Cabin',axis=1)
base = base.dropna()
colunas = base.dtypes[base.dtypes.values == 'object'].index
base = base.drop(colunas,axis=1)
```

```
# Para a base de teste
teste = pd.read_csv('titanic_test.csv')
teste = teste.drop('Cabin',axis=1)
teste = teste.dropna()
teste = teste.drop(colunas,axis=1)
```

Relembrando o que foi feito na última aula

Aqui nós já vamos começar o tratamento de dados. Vamos iniciar retirando colunas com alta cardinalidade e eliminando também a coluna Cabin pela alta cardinalidade e quantidade de valores vazios.

Etapa 1: Começando o tratamento de dados

```
# Importando novamente a base
base = pd.read_csv('titanic_train.csv')
base.head(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

Retirando colunas com alta cardinalidade

```
# Retirando as colunas
base = base.drop(['PassengerId','Name'],axis=1)
```

Eliminando a coluna Cabin pela alta cardinalidade e quantidade de valores vazios

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

Etapa 1 – Iniciando o tratamento de dados

Vamos tirar essas colunas, pois com uma alta cardinalidade o nosso modelo vai estar mais decorando do que de fato aprendendo.

Para o próximo passo você deve lembrar que temos algumas informações onde a idade é vazia, então também precisamos fazer o tratamento.

```
base['Age'].mean()
```

```
29.69911764705882
```

```
# Filtrando apenas Age vazio
base.loc[base.Age.isnull(),'Age']
```

```
5     NaN
17    NaN
19    NaN
26    NaN
28    NaN
      ..
859   NaN
863   NaN
868   NaN
878   NaN
888   NaN
Name: Age, Length: 177, dtype: float64
```

```
# Atribuindo a média para esses valores
base.loc[base.Age.isnull(),'Age'] = base['Age'].mean()
```

E só então apagar as linhas com valores vazios

```
# Eliminando todas as outras linhas com valores vazios
base = base.dropna()
```

Para conseguir usar esses dados, não podemos ter dados do tipo objeto, eliminando novamente

```
# Novamente eliminando os dados do tipo objeto
colunas = base.dtypes[base.dtypes.values == 'object'].index
base = base.drop(colunas,axis=1)
```

Inserindo a média de idade nos valores de idades vazios

Antes de excluir as linhas vazias da nossa base de dados vamos calcular a média das idades e atribuir esse valor onde temos as informações de idades vazias.

Isso é interessante, pois dessa forma vamos ter mais informações para trabalhar. Nesse caso a média é uma boa métrica para inserir esses dados, mas é possível quando você tenha dados muito altos pode ser que a mediana seja mais adequada.

Por isso é bom que você entenda bem os conceitos estatísticos para fazer essas substituições de forma que consiga trabalhar bem com esses dados.

Por fim nós vamos apagar as linhas com valores vazios e vamos eliminar as informações do tipo object, como já fizemos em aulas anteriores.

Vamos agora verificar as informações que nós temos e podemos fazer novamente a avaliação do modelo.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  889 non-null    int64
 1   Pclass    889 non-null    int64
 2   Age       889 non-null    float64
 3   SibSp     889 non-null    int64
 4   Parch     889 non-null    int64
 5   Fare      889 non-null    float64
dtypes: float64(2), int64(4)
memory usage: 48.6 KB
```

```
# Para a base de teste
teste = pd.read_csv('titanic_test.csv')
teste = teste.drop(['PassengerId','Name'],axis=1)
teste = teste.drop('Cabin',axis=1)
teste.loc[teste.Age.isnull(),'Age'] = teste.Age.mean()
teste = teste.dropna()
teste = teste.drop(colunas,axis=1)
```

Agora vamos avaliar o modelo com esses dados

Avaliando o modelo

Verificando as informações

Lembrando que todos os códigos para avaliação estão no final do código, então se clicar no link **Avaliando o modelo** você já vai direto para a parte final.

Você pode até anotar essas informações para ir verificando se o tratamento de dados está ou não melhorando suas métricas.

	A	B	C	D
1		Acurácia	Precisão	Recall
2	KNN	0.48338368580060426	0.40350877192982454	0.7244094488188977
3	Árvore	0.5861027190332326	0.47191011235955055	0.6614173228346457
4	RegLog	0.5861027190332326	0.4632352941176471	0.49606299212598426
5		0.5875299760191847	0.43243243243243246	0.42105263157894735
6		0.6067146282973621	0.461038961038961	0.46710526315789475
7		0.6282973621103117	0.48717948717948717	0.375
8				

Registrando os resultados de acurácia, precisão e recall para cada modelo

Dá só uma olhada como esse tratamento de dados melhorou algumas dessas métricas, é claro que não vamos conseguir melhorar tudo, mas podemos continuar fazendo nossas análises e utilizar o melhor modelo de acordo com a necessidade.

Como nos modelos nós só conseguimos analisar informações numéricas nós vamos inserir a informação de gênero na nossa base de dados, só que para isso vamos ter que transformar isso em números.

Então vamos utilizar a estrutura if para verificar se é “male” (masculino) e se for vamos atribuir 1, caso contrário vamos atribuir 0.

Assim conseguimos colocar na nossa base essas informações de gênero para analisarmos.

```
base = pd.read_csv('titanic_train.csv')
base.head(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

```
# Quais informações temos na nossa coluna de gênero?
base.Sex.value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```

Criando uma nova coluna apenas verificando se o valor é Male

```
# Criando uma função lambda para verificar essa informação
base['IsMale'] = base.Sex.apply(lambda x:1 if x == 'male' else 0)
```

```
# Usando o groupby para verificar a coluna criada
base.groupby(['Sex','IsMale'])['Sex'].count()
```

```
Sex     IsMale
female  0         314
male    1         577
Name: Sex, dtype: int64
```

Etapa 2 – Incluindo informação de gênero na base de dados

Feitos isso podemos fazer os mesmos tratamentos da etapa 1 e depois podemos avaliar os modelos com esses dados dessa maneira.

Fazendo os mesmos tratamentos da etapa 1

```
base = base.drop(['PassengerId','Name'],axis=1)
base = base.drop('Cabin',axis=1)
base.loc[base.Age.isnull(),'Age'] = base.Age.mean()
base = base.dropna()
base = base.drop(colunas,axis=1)
```

```
base.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  889 non-null    int64
 1   Pclass    889 non-null    int64
 2   Age       889 non-null    float64
 3   SibSp     889 non-null    int64
 4   Parch     889 non-null    int64
 5   Fare      889 non-null    float64
 6   IsMale    889 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 55.6 KB
```

```
# Para a base de teste
teste = pd.read_csv('titanic_test.csv')
teste['IsMale'] = teste['Sex'].apply(lambda x:1 if x == 'male' else 0)
teste = teste.drop(['PassengerId','Name'],axis=1)
teste = teste.drop('Cabin',axis=1)
teste.loc[teste.Age.isnull(),'Age'] = teste.Age.mean()
teste = teste.dropna()
teste = teste.drop(colunas,axis=1)
```

Agora vamos avaliar o modelo com esses dados

Avaliando o modelo

Repetindo os tratamentos da etapa 1

Para a terceira etapa nós vamos inserir a informação do porto de embarque dentro da base de dados, e aqui como não exista uma relação entre os valores vamos utilizar o **One Hot Encoding**.

Com isso vamos gerar 3 colunas, uma para cada porto, onde cada porto vai ser representado por uma dessas colunas, então só vamos ter o valor 1 em uma dessas colunas, nunca mais de um ao mesmo tempo.

Fazendo os mesmos tratamentos das etapas anteriores

```
base = base.drop(['PassengerId','Name'],axis=1)
base['IsMale'] = base['Sex'].apply(lambda x:1 if x == 'male' else 0)
base = base.drop('Cabin',axis=1)
base.loc[base.Age.isnull(),'Age'] = base.Age.mean()
base = base.dropna()
base = base.drop(colunas,axis=1)
```

```
# Para a base de teste
teste = pd.read_csv('titanic_test.csv')
teste['IsMale'] = teste['Sex'].apply(lambda x:1 if x == 'male' else 0)
teste = pd.concat([teste,pd.get_dummies(teste.Embarked)],axis=1)
teste = teste.drop(['PassengerId','Name'],axis=1)
teste = teste.drop('Cabin',axis=1)
teste.loc[teste.Age.isnull(),'Age'] = teste.Age.mean()
teste = teste.dropna()
teste = teste.drop(colunas,axis=1)
```

Agora vamos avaliar o modelo com esses dados

Avaliando o modelo

Tratando os dados

Depois podemos novamente avaliar os modelos com esses dados.

Para a última etapa nós vamos retirar algumas colunas da base para fazer nossas análises e verificar os resultados.

Etapa 4: Retirando colunas da base

```
# Vamos fazer todos os tratamentos que fizemos anteriormente
base = pd.read_csv('titanic_train.csv')
# base = pd.concat([base,pd.get_dummies(base.Embarked)],axis=1)
base = base.drop(['PassengerId','Name'],axis=1)
base['IsMale'] = base['Sex'].apply(lambda x:1 if x == 'male' else 0)
base = base.drop('Cabin',axis=1)
base.loc[base.Age.isnull(),'Age'] = base.Age.mean()
base = base.dropna()
base = base.drop(colunas,axis=1)
base.head(2)
```

	Survived	Pclass	Age	SibSp	Parch	Fare	IsMale
0	0	3	22.0	1	0	7.2500	1
1	1	1	38.0	1	0	71.2833	0

```
# Podemos retirar outras colunas da base
retirar = ['SibSp','Parch']
base = base.drop(retirar,axis=1)
```

```
# Para a base de teste
teste = pd.read_csv('titanic_test.csv')
teste['IsMale'] = teste['Sex'].apply(lambda x:1 if x == 'male' else 0)
# teste = pd.concat([teste,pd.get_dummies(teste.Embarked)],axis=1)
teste = teste.drop(['PassengerId','Name'],axis=1)
teste = teste.drop('Cabin',axis=1)
teste.loc[teste.Age.isnull(),'Age'] = teste.Age.mean()
teste = teste.dropna()
teste = teste.drop(colunas,axis=1)
teste = teste.drop(retirar,axis=1)
```

Agora vamos avaliar o modelo com esses dados

Avaliando o modelo

Etapa 4 – Retirando as colunas da base de dados

Claro que você vai analisar todas as etapas, pode até ir registrando os resultados de acurácia, precisão e recall para cada um dos modelos de classificação.

3	Árvore	0.5861027190332326	0.47191011235955055	0.6614173228346457
4	RegLog	0.5861027190332326	0.4632352941176471	0.49606299212598426
5	KNN	0.5875299760191847	0.43243243243243246	0.42105263157894735
6	Árvore	0.6067146282973621	0.461038961038961	0.46710526315789475
7	RegLog	0.6282973621103117	0.48717948717948717	0.375
8	KNN	0.6618705035971223	0.535031847133758	0.5526315789473685
9	Árvore	0.7913669064748201	0.7152317880794702	0.7105263157894737
10	RegLog	0.9448441247002398	0.9161290322580645	0.9342105263157895
11	KNN	0.5515587529976019	0.3939393939393939	0.4276315789473684
12	Árvore	0.5851318944844125	0.42758620689655175	0.40789473684210525
13	RegLog	0.5755395683453237	0.4260355029585799	0.47368421052631576
14	KNN	0.7410071942446043	0.6617647058823529	0.6592105263157894
15	Árvore	0.7577937649880095	0.6888888888888889	0.6118421052631579
16	RegLog	0.9328537170263789	0.8875	0.9342105263157895
17		0.6666666666666666	0.5424836601307189	0.5460526315789473
18		0.7961630695443646	0.7278911564625851	0.7039473684210527
19		0.9568345323741008	0.935064935064935	0.9473684210526315

Registro total de todas as métricas

Com isso nós temos que a etapa 4 levou a uma acurácia de 95%, uma precisão de 93% e um recall de 94%. Então se o cliente queria algo acima de 90% nós já podemos parar por aqui.

Você vai notar que cada etapa vai ter um resultado diferente, então você pode ir fazendo o seu tratamento até chegar nos valores que foram estipulados.

Veja que já tivemos uma precisão de 94% com outro tratamento de dados, por isso que vai depender do que você precisa para o projeto para não ter que ficar fazendo esses tratamentos infinitamente.

Se a precisão de 90% já é suficiente poderíamos já ter parado onde temos 94%, mas é importante que você entenda que pode ir fazendo diferentes tipos de tratamentos para obter diferentes resultados, até chegar no seu objetivo!

Aqui você nota que o nosso modelo inicial era de 58% de acurácia e nós conseguimos chegar em um modelo com 95% de acurácia.

Com isso você nota o quão importante é o tratamento de dados para melhorar o nosso modelo de classificação.

Aula 6 – Deploy do Modelo de Aprendizado de Máquinas

Hoje no Curso de Ciência de Dados Aula6 vamos te mostrar como fazer o deploy do modelo para que outras pessoas possam utilizá-lo!

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