Inteligência artificial:

Análise de jogadores de futebol americano

Prof: Thaís Gaudecio do Rêgo
Alunos: Claudio de Souza Brito
Matheus Arnaud Macambira Guedes

Motivação

Prever o sucesso (ou não) dos jogadores de futebol americano, usando como base de treinamento uma base de dados contendo jogadores veteranos, e cada coluna representa um dado importante de sua carreira na época universitária.

Base de dados

Foi escolhida uma base de dados que contém todos os jogadores da posição de quarterback que conseguiram sair do universitário e chegar ao profissional desde 1998 até o ano de 2019,com seus dados como jogadores universitários.Nesta base dados o número de jogos,a quantidade de passes completos,passes interceptados,jardas ganhas,touchdowns,tentativas de corrida,jardas corridas ganhas,touchdowns corridos,por qual time foram escolhidos, em quall conferencia jogaram,em qual universidade jogaram, a idade e o veredito se foram bons jogadores ou não.

Pré processamento

1. Correlação

Com a correlação podemos nos livrar de colunas redundantes, facilitando todas os cálculos que faremos posteriormente

Consideramos correlação maior que 0,75 para eliminar as colunas

Pré processamento

1. Correlação

Round	Pick	Age	GamesPlayed	Completions	Attempts	Yards	Touchdowns	Interceptions	RushAttempts	RushYards	RushTouchdowns	V
000000	0.989429	0.214269	-0.063742	-0.245640	-0.209419	-0.250641	-0.266463	-0.041335	-0.164485	-0.110784	-0.215569	-0.4
989429	1.000000	0.196506	-0.067464	-0.249594	-0.220593	-0.259578	-0.278116	-0.030172	-0.166389	-0.114716	-0.217578	-0.4
214269	0.196506	1.000000	0.099885	0.023472	0.010321	0.012922	0.015743	0.050814	-0.123096	-0.150614	-0.181505	-0.2
063742	-0.067464	0.099885	1.000000	0.566399	0.660696	0.608379	0.548838	0.495500	0.383462	0.155098	0.237879	-0.0
245640	-0.249594	0.023472	0.566399	1.000000	0.904562	0.842364	0.830586	0.645719	0.058315	-0.196245	-0.021719	0.0
209419	-0.220593	0.010321	0.660696	0.904562	1.000000	0.866614	0.812643	0.760709	0.180260	-0.123591	0.040615	0.0
250641	-0.259578	0.012922	0.608379	0.842364	0.866614	1.000000	0.834884	0.547826	0.159135	-0.078864	0.091726	0.0
266463	-0.278116	0.015743	0.548838	0.830586	0.812643	0.834884	1.000000	0.475991	0.117356	-0.053413	0.072985	0.0
041335	-0.030172	0.050814	0.495500	0.645719	0.760709	0.547826	0.475991	1.000000	0.103543	-0.194023	-0.066070	-0.0
164485	-0.166389	-0.123096	0.383462	0.058315	0.180260	0.159135	0.117356	0.103543	1.000000	0.876245	0.875311	0.0
110784	-0.114716	-0.150614	0.155098	-0.196245	-0.123591	-0.078864	-0.053413	-0.194023	0.876245	1.000000	0.898043	0.0
215569	-0.217578	-0.181505	0.237879	-0.021719	0.040615	0.091726	0.072985	-0.066070	0.875311	0.898043	1.000000	0.0
472794	-0.464406	-0.289277	-0.041002	0.096577	0.043444	0.063542	0.073724	-0.031312	0.028938	0.044790	0.061384	1.0

Pré processamento

2. Deletando instâncias duplicadas ou nulas

“Limpar” a base de dados é essencial para que não atrapalhe nossa análise.

Não haviam dados nulos nem duplicados

```
data.isnull().sum()
```

```
Out[3]: Round          0
        Age            0
        GamesPlayed    0
        Completions    0
        RushAttempts   0
        Player         0
        College        0
        Conference     0
        Team           0
        Verdict        0
        dtype: int64
```

```
In [5]: data.duplicated().sum()

Out[5]: 0
```

Pré processamento

2. Deletando instâncias duplicadas ou nulas

“Limpar” a base de dados é essencial para que não atrapalhe nossa análise.

Não haviam dados nulos nem duplicados

```
data.isnull().sum()
```

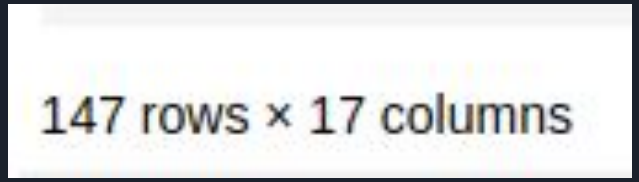
```
Out[3]: Round          0
        Age            0
        GamesPlayed    0
        Completions    0
        RushAttempts   0
        Player         0
        College        0
        Conference     0
        Team           0
        Verdict        0
        dtype: int64
```

```
In [5]: data.duplicated().sum()

Out[5]: 0
```

Problemas encontrado

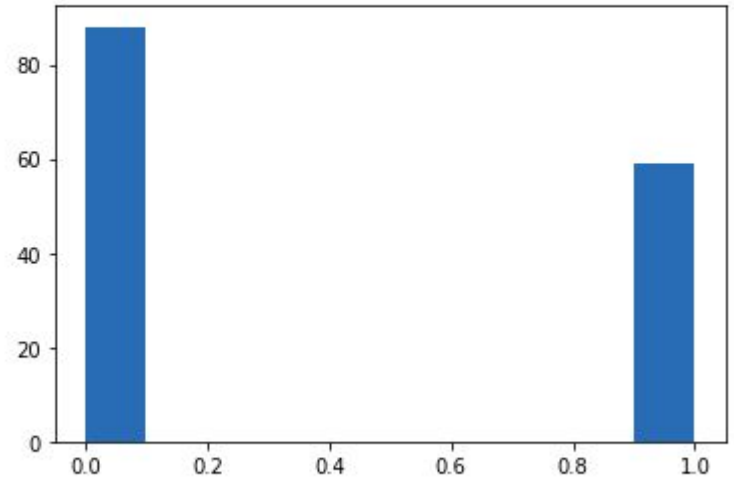
Pequeno número de amostras



Problemas encontrado

Classes desbalanceadas

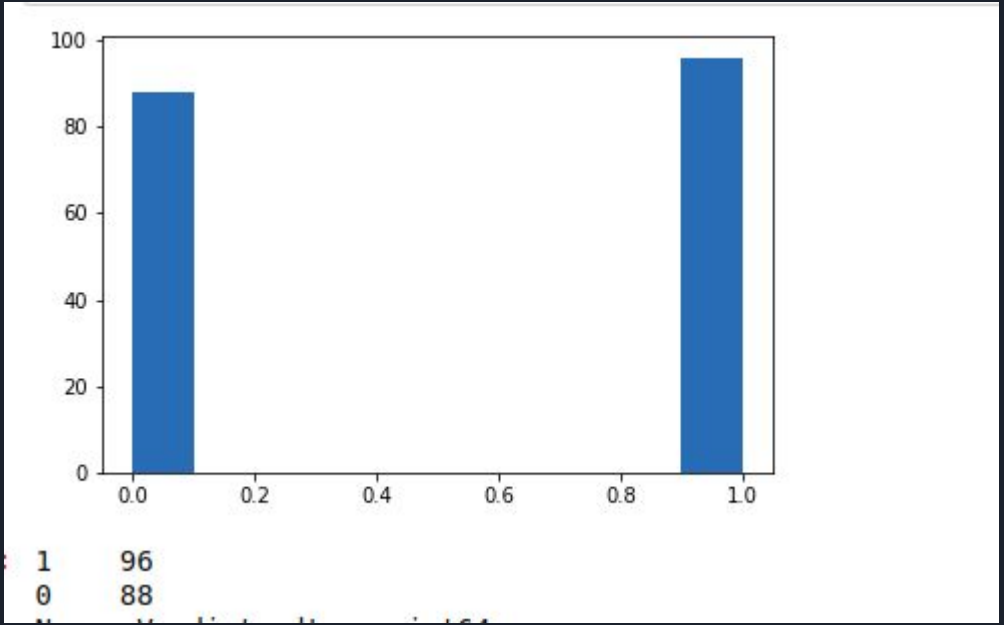
```
In [4]: plt.hist(data['Verdict'])
        plt.show()
        data.Verdict.value_counts()
```




```
Out[4]: 0    88
        1    59
```

Solução

Oversampling: ADASYN



Processamento

1. Classificação: KNN

O método dos k vizinhos próximos nos permite classificar instâncias em grupos, usando a hipótese de que instâncias parecidas ficam próximas umas das outras no espaço de dispersão de dados.

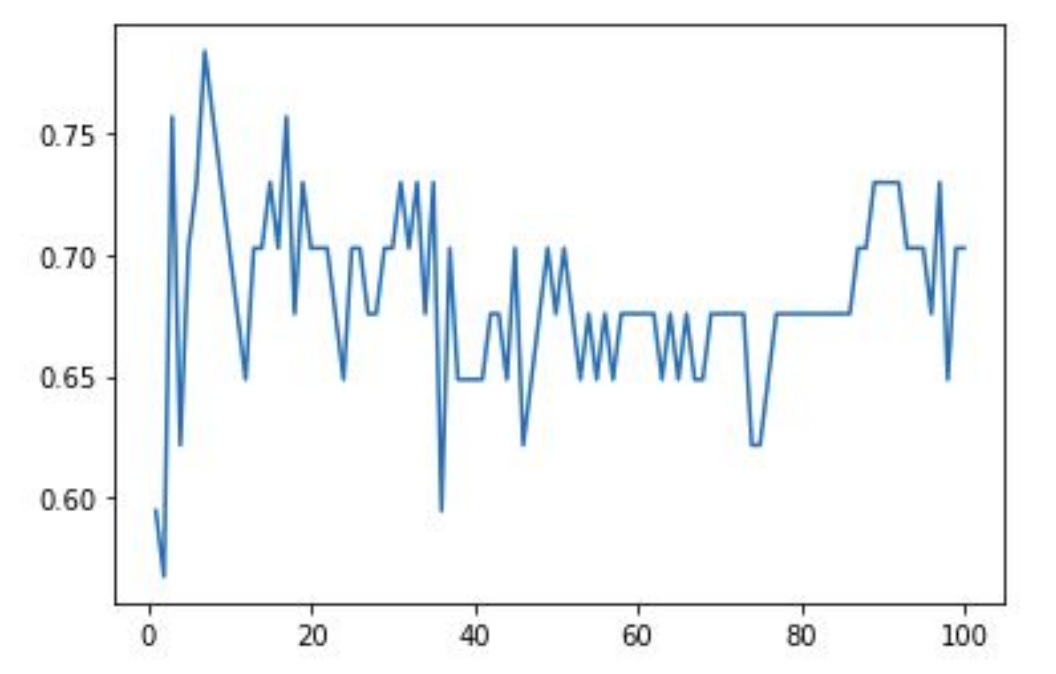
Relação treino/teste: 80/20

Sem saber qual número usar para “K”, testamos de 1 a 100 para ver qual o melhor resultado.

Processamento

1. Classificação: KNN

   Y: acurácia X: k



Processamento

1. Classificação: RandomForestClassifier (RFC)

Usando a função: GridSearchCV podemos escolher vários parâmetros para ser testados x vezes (validação cruzada). Entre os parâmetros está o algoritmo escolhido, desta vez escolhemos o RFC.

```
param_grid = { 'max_features': ['sqrt', 'log2'],
               'criterion': ['gini', 'entropy'],
               'max_depth': [5, 10, 15, 20]}

clf = GridSearchCV(RandomForestClassifier(), param_grid, cv=10, scoring="accuracy", return_train_score=False)
```

Processamento

1. Classificação: RFC

```
Best parameters combination: {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt'}

Test score: 75.676%

Matriz de Confusão:
[[14  4]
 [ 5 14]]
```

Processamento

2. Regressão: Linear

   Separando treino/teste em 80/20.

   Validação cruzada??

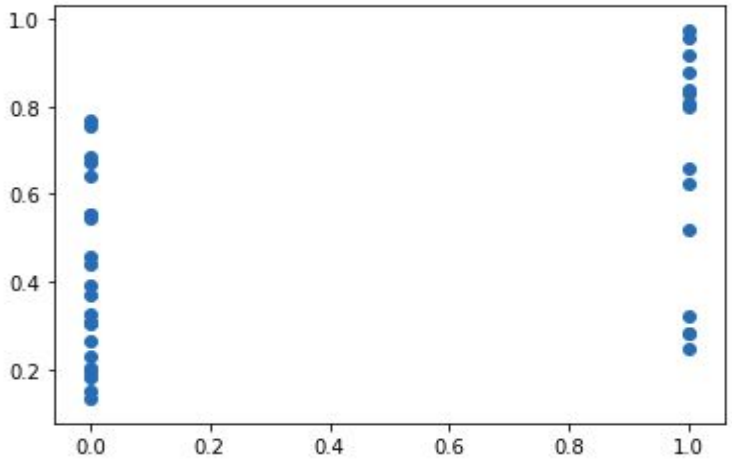
Processamento

2. Regressão: Linear

X: saídas originais Y: saídas previstas

```
MAE: 0.3840591154804546
MSE: 0.1991620298089851
RMSE: 0.44627573293759215
```

Out[11]: <matplotlib.collections.PathCollection at 0x7f86f1669048>



Clusterização

3. PCA

Queremos saber se nosso algoritmo funciona bem para classificação não supervisionada também, então faremos clusterização.

mas antes é necessário reduzir a dimensão da nossa base de dados, então faremos PCA

Clusterização

3. PCA

	principal component 1	principal component 2	Verdict
0	27.017393	61.520546	0
1	-358.376072	-8.154570	0
2	-9.971165	-128.049515	0
3	-365.830566	-122.479999	0
4	-160.708953	-3.783513	0
...	...	...	...
179	16.581377	73.029749	1
180	-216.651007	-96.143125	1
181	-420.686608	-106.222939	1
182	9.724952	389.268005	1
183	-258.450665	-36.965039	1

184 rows × 3 columns

Clusterização

3. K-means

   Setando 2 centróides aleatoriamente