FEUP-IART T2-4C

Gustavo Speranzi Tosi Tavares
MIEIC
FEUP
Porto, Portugal
up201700129@fe.up.pt

Daniel Gazola Bradaschia
MIEIC
FEUP
Porto, Portugal
up201700494@fe.up.pt

Abstract
Como o sucesso de uma equipe depende do desempenho de seus individuos, a avaliação do desempenho de um jogador tornou-se um tópico de pesquisa essecial para uma equipe. Neste artigo, avaliamos diferentes algoritmos de aprendizado de máquina, usando dados de aproximadamente 20.000 jogos de 11 ligas europeias durante um período de 8 temporadas, bem como, classificações de jogadores da FIFA, a fim de previr o desempenho de um jogador, assim como explorar os resultados dos algoritmos utilizados.

Keywords— Sports analytics · Data mining · Player valuation · Supervised Learning · Regression

I. Introdução

Neste trabalho tem como proposito, a aplicação de conceitos de aprendizagem supervisionada do tipo regressivo, fazendo uso de uma extensa base de dados de futebol da liga européia. Pretende-se assim, prever o desempenho de um jogador, fazendo uso dos dados disponibiliados.

II. Descrição do Problema

A. *Análise do Problema*

Assim como explicado anteriormente, o tema tem como intuito fazer uso de ferramentas de Machine Learning para analisar uma quantidade extensa de dados através do dataset fornecido e por fim fazer previsões precisas do desempenho de um jogador, mas fazendo uso da múltiplos algoritmos da biblioteca Machine Learning scikit-learn, sendo os algoritmos escolhidos: Árvores de Decisão, SGDRegressor, K-Nearest, Redes Neuronais.

B. *Explicação do Dataset*

O dataset fornecido foi da liga de futebol européia de Hugo Mathien, onde é disponibilizado dados sobre temporadas da liga européia entre 2008 e 2016, com informações detalhadas sobre 10000+ jogadores e 250000+ partidas, fazendo uso também de informações da série de jogo FIFA.

III. Abordagem

A. *Ferramentas*

Na implementação do projeto, utilizamos da linguagem Python, as bibliotecas NumPy, pandas, Matplotlib, Machine Learning scikit-learn e as plataformas IPython e Jupyter Notebook.

B. *Desenvolvimento*

A Aplicação pode ser dividida em duas estapas:

- Configuração dos dados
- Aplicação dos algoritmos

1) *Configuração dos dados*

Nesta etapa, destina-se a leitura da base de dados, o pré-processamento e a transformação dos dados.

Após o código se conectar a base de dados através da linha de comando.

```
with sqlite3.connect('../soccer/database.sqlite') as con:
Player_Attributes = pd.read_sql_query("SELECT * from Player_Attributes", con)
```

A etapa seguinte é efetuar a limpeza dos dados, inicialmente através de remover colunas irrelevantes, como data, ids e o overall rating, que é nosso objetivo final a aplicação ou que apresentem valores nulos através da seguinte passagem.

```
Player_Attributes.dropna(inplace=True)
Player_Attributes.drop(['id', 'player_fifa_api_id', 'player_api_id', 'date'], axis = 1, inplace = True)
overall_rating = Player_Attributes['overall_rating']
features = Player_Attributes.drop('overall_rating', axis = 1)
```

Em seguida, é necessário que as colunas com valores não númericos (prefered foot, attacking work rate e defensive work rate), decidimos converter seus valores em valores numéricos através da seguinte linha, mas outra opção seria elimina-las junto as categorias anteriores.

```
features = pd.get_dummies(features)
```

Finalmente, para encerrar essa etapa, fazemos uso do MinMaxScaler para uniformizar os dados, com o intuito de aumentar a precisão adquirida dos valores.

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
scaled_features = min_max_scaler.fit_transform(features)
```

2) *Aplicação dos algoritmos*

Nesta etapa que iremos aplicar os 4 diferentes algoritmos selecionados sobre os dados configurados na etapa anterior e

efetuar a previsão através fazendo uso dos sets de treino e teste através do código a seguir. Relembrando que os algoritmos em questão são Árvores de Decisão, SGDRegressor, K-Nearest, Redes Neuronais.

```
reg1 = tree.DecisionTreeClassifier()
reg2 = linear_model.SGDRegressor()
reg3 = KNeighborsRegressor(n_neighbors=2)
reg4 = MLPRegressor(random_state=1, max_iter=500)

regs = {reg1:"DecisionTree", reg2:"SGDRegressor",
reg3:"KNeighbors", reg4:"NeuralNetwork"}

for key in regs:
    t0 = time()
    X_train, X_test, y_train, y_test =
train_test_split(scaled_features, overall_rating, test_size=0.25,
random_state=0)

    print ("--------------------")
    print (regs[key])
    print ("--------------------")

    t1 = time()
    key.fit(X_train, y_train)
    print ("Time taken to train the model: {}".format(time()-t1))

    t2 = time()
    pred_test = key.predict(X_test)
    pred_train = key.predict(X_train)
    print ("Time taken to predict the model: {}".format(time()-
t2))

    t3 = time()
    print ("r2 score of this model on testing set is:
{}".format(r2_score(y_test, pred_test)))
    print ("r2 score of this model on training set is:
{}".format(r2_score(y_train, pred_train)))
```

IV. Experimentação

A. *Comparação dos algoritmos*

Nesta primeira experiência efetuamos a comparação dos algoritmos de acordo com sua precisão dos resultados e o tempo necessário para efetuar os cálculos.

DecisionTree

Time taken to train the model: 7.420112371444702
Time taken to predict the model: 0.21271991729736328
r2 score of this model on testing set is: 0.9497573508853848
r2 score of this model on training set is: 0.9988644730721816

SGDRegressor

Time taken to train the model: 2.0376570224761963
Time taken to predict the model: 0.015960216522216797
r2 score of this model on testing set is: 0.8429320134680672
r2 score of this model on training set is: 0.8445771303753988

KNeighbors

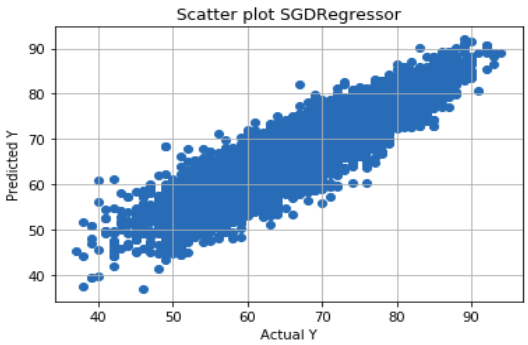
Time taken to train the model: 12.83634877204895
Time taken to predict the model: 722.3275904655457
r2 score of this model on testing set is: 0.9557633414780853
r2 score of this model on training set is: 0.9891838192633062

NeuralNetwork

Time taken to train the model: 853.534113407135
Time taken to predict the model: 5.250151634216309
r2 score of this model on testing set is: 0.9559526964416791
r2 score of this model on training set is: 0.9579127454206207

B. *Precisão*

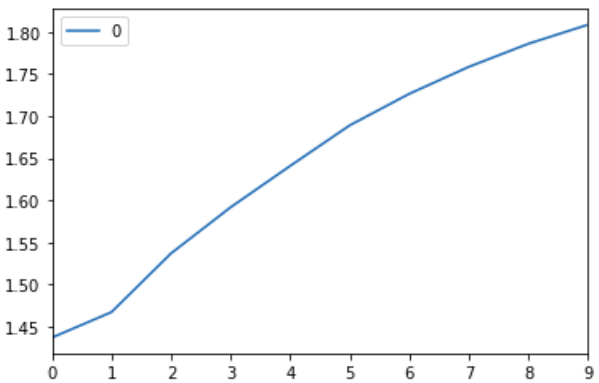
Em seguida pode-se avaliar um gráfico de acordo com o algoritmo SGDRegressor, que revela a precisão dos resultados de acordo com os resultados reais.



C. *RSME*

Em seguida foi efetuado testes de RSME para o algoritmo de K-Nearest, de acordo com o valor k que simboliza o número de vizinhos próximos. Os cálculos param após a décima iteração, pois o RSME passa a se estabilizar em seguida.

RMSE value for k= 1 is: 1.4371127607135425
RMSE value for k= 2 is: 1.4674191600658149
RMSE value for k= 3 is: 1.5371942246708927
RMSE value for k= 4 is: 1.5919015401176657
RMSE value for k= 5 is: 1.6410105998357583
RMSE value for k= 6 is: 1.6891998468890845
RMSE value for k= 7 is: 1.7266537186638116
RMSE value for k= 8 is: 1.7586972550592932
RMSE value for k= 9 is: 1.7860959172709656
RMSE value for k= 10 is: 1.8086762098454865



V. Conclusão

No âmbito da matéria de inteligência artificial o projeto cumpriu o objetivo de criar modelos de regressão, fazendo uso de algoritmos de aprendizagem supervisionada e um dataset específico.

O trabalho foi finalizado com sucesso, mas com consciência que mais poderia ter sido feito em relação as experiências feitas quanto a demonstração mais detalhada dos resultados.

Participação

- Daniel Gazola Bradaschia 60%
- Gustavo Speranzini Tosi Tavares 40%

Referencias

[1] *Efezino Erome-Utunedi* https://nycdatascience.com/blog/student-works/analyzing-predicting-european-soccer-match-outcomes/

[2] Prakhar Rati https://www.kaggle.com/prakharrathi25/european-soccer-regression-analysis/notebook

[3] Hugo Mathien https://www.kaggle.com/prakharrathi25/european-soccer-regression-analysis/notebook