IART Trabalho 2

Aprensentação

28 maio 2020

Francisco Gonçalves 201704790
Luís Ramos 201706253
Martim Silva 201705205

3MIEIC_04 (Tema 3C - Previsão de resultados de futebol)

Sumário

Neste projeto vamos tentar conceber um modelo de previsão, capaz de adivinhar com relativa exatidão o resultado dum jogo de futebol relativo à equipa da casa (vitória, empate ou derrota) isto é., prever o desfecho de jogos de futebol.

Todos os modelos serão baseados no dataset fornecido que conta com dados básicos de jogos, estatísticas de jogadores do FIFA e dados de *bookkeeper*.

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Recursos

European Soccer Database
Football Data Analysis
Data Analysis and Machine Learning Projects
Match Outcome Prediction in Football
European Soccer Database Supplementary (XML Events to CSV)

A deep learning framework for football match prediction
Predicting Football Match Outcome using Machine Learning: Football Match prediction using machine learning algorithms in jupyter notebook
(PDF) Football Result Prediction by Deep Learning Algorithms
Predicting Football Results Using Machine Learning Techniques
A machine learning framework for sport result prediction

Ferramentas, *frameworks* e/ou algoritmos usados

Utilizámos Jupyter Notebook para escrever, visualizar, descrever todo o código.

No que toca a algoritmos, desenvolvemos os modelos de Aprendizagem Supervisionada: K-Nearest Neighbours, Support Vector Machines, Deep Neural Networks, Gradient Boosting e Naive Bayes através do uso de Scikit-learn e Keras do Tensorflow.

Relativamente a ferramentas, recorremos à biblioteca pandas para extração de dados, matplotlib para visualização dos mesmos e numpy para uso comum de cálculos.

Análise dos dados

Ao ler acerca de trabalhos feitos com os dados fornecidos (European Soccer Database) foi possível verificar que estes não se encontravam na melhor condição a fim de serem usados para machine learning. Isto devido a ter bastantes colunas com valores NULL e outras até com o conteúdo em XML.

goal		shoton		shotoff		foulcommit		card		cross		corner		possession	
[null]	45%	[null]	45%	[null]	45%	[null]	45%	[null]	45%	[null]	45%	[null]	45%	[null]	45%
<goal />	4%	<shoton />	22%	<shotoff />	22%	<foulcommit />	22%	<card />	2%	<cross />	22%	<corner />	22%	<possession />	22%
Other (13224)	51%	Other (8463)	33%	Other (8463)	33%	Other (8465)	33%	Other (13776)	53%	Other (8465)	33%	Other (8464)	33%	Other (8419)	32%

Apesar de este *stepback* conseguimos encontrar um dataset (European Soccer Database Supplementary (XML Events to CSV)) que já tinha traduzido estes dados para ficheiros .csv que são bastante mais fáceis de tratar.

Adição de mais features

Nesta parte foram feitos os processamentos relativos aos dados provenientes das colunas em XML. Nomeadamente os remates, a posse de bola, os cantos e os cruzamentos. Para isto tivemos de cruzar os dados visto que provêm de um dataset diferente do inicial.

Ao juntarmos os dados aos jogos foi possível ver que algumas das equipas tinham poucos ou nenhuns dados relativos aos jogos em que participam. Sendo que tinhamos cerca de 1458 equipas:

avg_shots
997.000000
10.579949
1.808530
3.000000
9.470588
10.428571
11.376812
16.000000

avg_corners
997.000000
10.807166
1.916837
3.000000
9.700000
10.591837
11.526316
21.000000

avg_crosses
997.000000
15.888237
2.815376
1.500000
14.358974
15.750000
17.782895
27.000000

possession
178.000000
4728.651685
4268.085611
38.000000
1650.750000
3690.500000
6623.500000
17221.000000

Filtragem e escolha dos dados

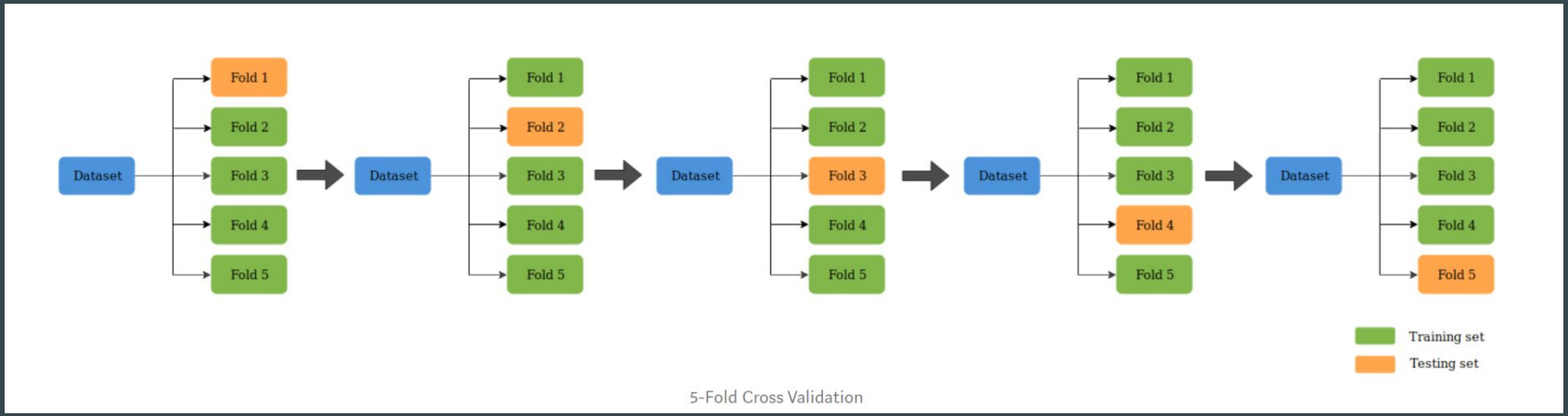
Começamos por eliminar as colunas dos jogos que não seriam úteis e de seguida, limpando os dados correspondentes ao jogos, retirando as informações relativas às posições iniciais dos jogadores, bem como a maior parte das informações relativas aos *bookkeepers*.

Após esta filtragem inicial é possível ainda ver que muitas colunas apresentam alguns valores a Nan. Com isto tivemos de utilizar a técnica de *mean imputation* para evitar ter de apagar demasiadas features que serão enviadas como input aos nossos modelos.

Procedemos também à normalização dos dados de entrada de forma a obter melhores resultados nos modelos utilizados.

Classificação

- KNN | Decision Tree | SVC | Naive Bayes | Gradient Boosting | Neural Network | Deep Neural Network
- K-Fold Cross Validation usado em vez de train_test_split

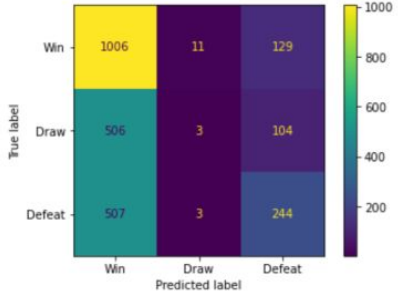


5-Fold Cross Validation

Análise das matrizes de confusão

KNN

```
clf = KNeighborsClassifier(n_neighbors=100)
train_predict(clf, features, outcomes)
```

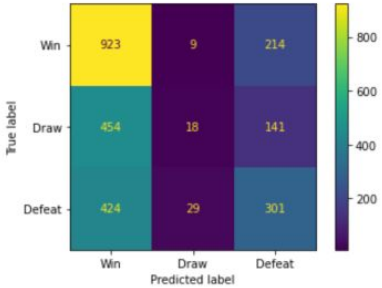


	precision	recall	f1-score	support
Win	0.51	0.32	0.40	754
Draw	0.18	0.00	0.01	613
Defeat	0.50	0.88	0.64	1146
accuracy			0.50	2513
macro avg	0.40	0.40	0.35	2513
weighted avg	0.42	0.50	0.41	2513

```
Accuracy:  0.4986072423398329
Recall:  0.4021124474336711
Precision:  0.395422485035644
F1 Score:  0.34721749376347616
```

Decision Tree

```
clf = DecisionTreeClassifier(random_state=0, criterion=
train_predict(clf, features, outcomes)
```

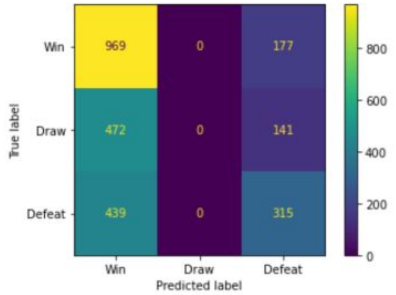


	precision	recall	f1-score	support
Win	0.46	0.40	0.43	754
Draw	0.32	0.03	0.05	613
Defeat	0.51	0.81	0.63	1146
accuracy			0.49	2513
macro avg	0.43	0.41	0.37	2513
weighted avg	0.45	0.49	0.43	2513

```
Accuracy:  0.4942300039793076
Recall:  0.41132605028715274
Precision:  0.4309210314182146
F1 Score:  0.3690539141134166
```

SVC

```
clf = SVC(coef0=5, kernel='poly')
train_predict(clf, features, outcomes)
```



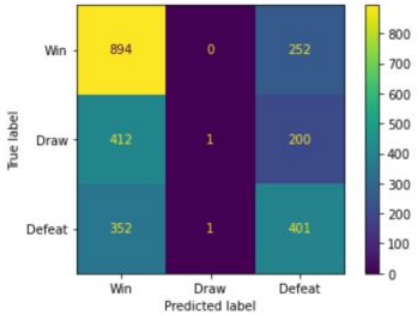
	precision	recall	f1-score	support
Win	0.50	0.42	0.45	754
Draw	0.00	0.00	0.00	613
Defeat	0.52	0.85	0.64	1146
accuracy			0.51	2513
macro avg	0.34	0.42	0.36	2513
weighted avg	0.38	0.51	0.43	2513

```
Accuracy:  0.5109430959013131
Recall:  0.4211072071696733
Precision:  0.3376852878894827
F1 Score:  0.36488905810779065
```

Análise das matrizes de confusão

Naive Bayes

```
clf = GaussianNB(var_smoothing=1.1)
train_predict(clf, features, outcomes)
```

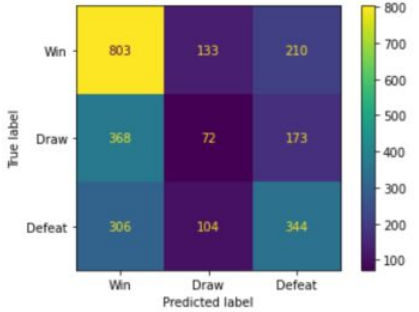


	precision	recall	f1-score	support
Win	0.47	0.53	0.50	754
Draw	0.50	0.00	0.00	613
Defeat	0.54	0.78	0.64	1146
accuracy			0.52	2513
macro avg	0.50	0.44	0.38	2513
weighted avg	0.51	0.52	0.44	2513

```
Accuracy:  0.5157182650218862
Recall:  0.43785542404632843
Precision:  0.5031031233457354
F1 Score:  0.3799930337460173
```

Gradient Boosting

```
clf = XGBClassifier(max_depth=20)
train_predict(clf, features, outcomes)
```



	precision	recall	f1-score	support
Win	0.47	0.46	0.46	754
Draw	0.23	0.12	0.16	613
Defeat	0.54	0.70	0.61	1146
accuracy			0.49	2513
macro avg	0.42	0.42	0.41	2513
weighted avg	0.45	0.49	0.46	2513

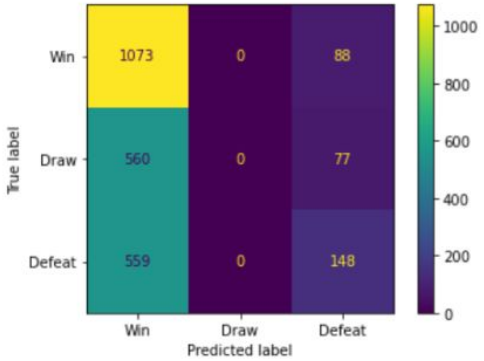
```
Accuracy:  0.4850775964982093
Recall:  0.42479554689740384
Precision:  0.41661891694002673
F1 Score:  0.4110030704913901
```

Análise das matrizes de confusão

Neural Network

```
visible = Input(shape=(features.shape[1],))
hidden = Dense(100, activation='relu')(visible)
output = Dense(3, activation='softmax')(hidden)

clf = Model(inputs=visible, outputs=output)
print(clf.summary())

from keras import metrics
from keras import losses
from keras import optimizers

clf.compile(optimizer=optimizers.Adam(),
              loss=losses.CategoricalCrossentropy(),
              metrics=[metrics.Precision(), metrics.Recall()])

train_predict_nn(clf, features, outcomes)
```

```
Model: "model_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_17 (InputLayer)        (None, 60)                0
_________________________________________________________________
dense_72 (Dense)             (None, 100)               6100
_________________________________________________________________
dense_73 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,403
Trainable params: 6,403
Non-trainable params: 0
_________________________________________________________________
```



	precision	recall	f1-score	support
Win	0.47	0.21	0.29	707
Draw	0.00	0.00	0.00	637
Defeat	0.49	0.92	0.64	1161
accuracy			0.49	2505
macro avg	0.32	0.38	0.31	2505
weighted avg	0.36	0.49	0.38	2505

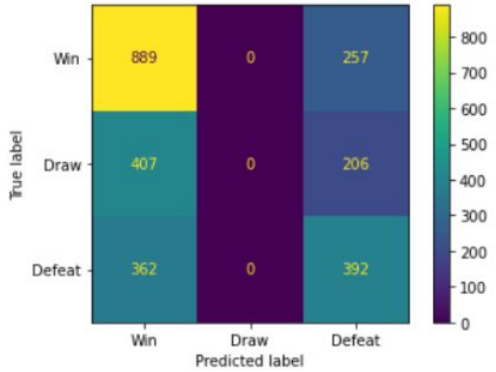
```
Accuracy:  0.4874251497005988
Recall:  0.37784616409223054
Precision:  0.32078358324976874
F1 Score:  0.31007331255397075
```

Análise das matrizes de confusão

Deep Neural Network

```
visible = Input(shape=(features.shape[1],))
hidden1 = Dense(500, activation='relu')(visible)
hidden2 = Dense(100, activation='relu')(hidden1)
hidden3 = Dense(50, activation='relu')(hidden2)
hidden4 = Dense(20, activation='relu')(hidden3)
output = Dense(3, activation='softmax')(hidden4)

clf = Model(inputs=visible, outputs=output)
print(clf.summary())

from keras import metrics
from keras import losses
from keras import optimizers

clf.compile(optimizer=optimizers.Adam(),
              loss=losses.CategoricalCrossentropy(),
              metrics=[metrics.Precision(), metrics.Recall()])

train_predict_nn(clf, features, outcomes)
```

```
Model: "model_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        (None, 60)                0
_________________________________________________________________
dense_58 (Dense)             (None, 500)               30500
_________________________________________________________________
dense_59 (Dense)             (None, 100)               50100
_________________________________________________________________
dense_60 (Dense)             (None, 50)                5050
_________________________________________________________________
dense_61 (Dense)             (None, 20)                1020
_________________________________________________________________
dense_62 (Dense)             (None, 3)                 63
=================================================================
Total params: 86,733
Trainable params: 86,733
Non-trainable params: 0
_________________________________________________________________
```



	precision	recall	f1-score	support
Win	0.46	0.52	0.49	754
Draw	0.00	0.00	0.00	613
Defeat	0.54	0.78	0.63	1146
accuracy			0.51	2513
macro avg	0.33	0.43	0.37	2513
weighted avg	0.38	0.51	0.44	2513

```
Accuracy:  0.5097493036211699
Recall:  0.43187853650030944
Precision:  0.33155590356403003
F1 Score:  0.37378443946571344
```

Escolha de medidas

Para analisar e avaliar os modelos foi necessário escolher uma medida de comparação. A exatidão (*accuracy*) apenas é útil quando estamos perante um *dataset* equilibrado, que não é o caso. Sendo assim considerámos apenas as outras 3 medidas possíveis: *recall*, precisão (*precision*) or *f-measure*.(*f1-score*).

Como uma previsão está associado a um custo (custo da aposta), a medida mais interessante é a precisão. Assim, o nosso objetivo é tentar obter a percentagem máxima de verdadeiros positivos que são classificados corretamente pelos nossos modelos.

Escolha do modelo

Tendo em conta o que foi dito, é de notar que todos os modelos obtiveram resultados positivos, sendo que alguns conseguem alcançar estes resultados de uma forma mais consistente.

Temos que ter em especial atenção aos algoritmos: *K-Nearest-Neighbours*, *Decision Tree, Naive Bayes*, *Gradient Boosting* que obtiveram sempre os melhores resultados de todos os modelos com uma *precision* aproximada de 47%.

Conclusão

Exploramos muitos conceitos que estavam pouco aprofundados, nos quais tínhamos apenas uma noção teórica. Aprendemos a importância da análise do *dataset*, e caso se verifique má usabilidade, a necessidade de fazer uma depuração deste.

Em suma, foi proveitoso fazer este trabalho, porque representou o primeiro contacto prático com *Machine Learning*. Tivemos a possibilidade de aprender por nós próprios, aprendendo com erros no processo e aumentar o interesse nesta área, sendo isso um estigma para investigar mais sobre a mesma no futuro.