Web Intelligence

Alexandra Pert

West University of Timișoara
Faculty of Mathematics and Computer Science

Thursday 20th February, 2020

Overview

Motivation & Problem

Motivation:
this project was constructed in order to find the winner of a future tennis championship. This can be useful at:

1. betting on a player and increase the chances at winning at sports betting;
2. helping sports clothing companies to know ahead of time which players have more chances at winning the championship and sponsor them for a good advertisement

Motivation & Problem

Motivation:
this project was constructed in order to find the winner of a future tennis championship. This can be useful at:

1. betting on a player and increase the chances at winning at sports betting;
2. helping sports clothing companies to know ahead of time which players have more chances at winning the championship and sponsor them for a good advertisement

Problem:
the project was modeled in Python programming language to complete the following task: choose the best classifier for predicting the winner of the Australian Open 2019 with data sets from excel files downloaded from:
`http://tennis-data.co.uk/alldata.php`

Components & General Idea

Components :

- tennis folders:
  - 2015-2019.xlsx for training our algorithm (with the 2019.xlsx having the data only until the Australia Open begins)
  - a partial 2019.xlsx containing only the Australian Open matches;
- utils.py: containing the functions used for the data processing part;
- main.ipynb: containing the prediction models;

General Idea : use data from all the tennis competitions that took place from 2015 to 2019 (until the Australian Open), fit them into 2 models that were implemented using the Python sklearn library :

- RandomForestClassifier;
- LogisticalRegression;

and choose the one with the best score for simulating the matches for Australian Open 2019.

Data Handling

1. The first step is to prepare the data for the models by choosing the X and Y parts of the model

In [9]:
```
#the relevant features for training
features = [
    "Tournament","Court",
    "Surface","Round",
    "Best of","Series",
    "P1Rank","P2Rank",
    "P1_Experince","P2_Experince",
    "P1_W/L","P2_W/L",
    "P1Pts","P2Pts",
    "AvgP1","AvgP2",
]
```

In [10]:
```
# Split the test and train data (our x and y)
X_train, X_test, y_train, y_test = train_test_split(matches_data[features], matches_data["P1_won"], test_size = TEST_SIZE)
print(f"Nr of training data:{len(X_train)}")
print(f"Nr of testing data:{len(X_test)}")
```

```
Nr of training data:7894
Nr of testing data:2632
```

Data Handling

1. The first step is to prepare the data for the models by choosing the X and Y parts of the model

In [9]:
```
#the relevant features for training
features = [
    "Tournament","Court",
    "Surface","Round",
    "Best of","Series",
    "P1Rank","P2Rank",
    "P1_Experince","P2_Experince",
    "P1_W/L","P2_W/L",
    "P1Pts","P2Pts",
    "AvgP1","AvgP2",
]
```

In [10]:
```
# Split the test and train data (our x and y)
X_train, X_test, y_train, y_test = train_test_split(matches_data[features], matches_data["P1_won"], test_size = TEST_SIZE)
print(f"Nr of training data:{len(X_train)}")
print(f"Nr of testing data:{len(X_test)}")
```

```
Nr of training data:7894
Nr of testing data:2632
```

2. Initially the data set was organised under the winner/loser format, but it was changed to the P1/P2 format and added an additional column(P1_won) that can signal if P1 or P2 won (True/False for the P1 column)
3. The P1_won is used as the Y for the models.

Data Handling

The X of the model will consist of some initial features found in the data set plus two custom features added further.

Existing features used from the data set :

- Tournament: the match tournament;
- Court: the match place(outdoor/indoor);
- Surface: the court surface(hard/clay/grass);
- Round: the hierarchical round;
- Best of: number of played sets;
- Series: the series of the match;
- P1Rank: the rank of P1 at the time of the match;
- P2Rank: the rank of P2 at the time of the match;
- P1Pts: the number of points for P1 at the time of the match;
- P2Pts: the number of points for P2 at the time of the match;
- AvgP1 betting score for P1;
- AvgP2 betting score for P2.

Data Handling

The custom features added are:

- P1_Experience;
- P2_Experience;
- P1_W/L;
- P2_W/L;

Data Handling

The custom features added are:

- P1_Experience;
- P2_Experience;
- P1_W/L;
- P2_W/L;

The experience is represented by the number of matches played by the player until the current match.

Data Handling

The custom features added are:

- P1_Experience;
- P2_Experience;
- P1_W/L;
- P2_W/L;

The experience is represented by the number of matches played by the player until the current match.

The W/L ratio is represented by: $\frac{Nr\_Loses}{Nr\_Wins} * 100$, where

Data Handling

The custom features added are:

- P1_Experience;
- P2_Experience;
- P1_W/L;
- P2_W/L;

The experience is represented by the number of matches played by the player until the current match.

The W/L ratio is represented by: $\frac{Nr\_Loses}{Nr\_Wins} * 100$, where

Nr_Loses is the number of loses the player has until the match date

Data Handling

The custom features added are:

- P1_Experience;
- P2_Experience;
- P1_W/L;
- P2_W/L;

The experience is represented by the number of matches played by the player until the current match.

The W/L ratio is represented by: $\frac{Nr\_Loses}{Nr\_Wins} * 100$, where

Nr_Loses is the number of loses the player has until the match date

Nr_Wins the total number of wins until the match date.

Data Handling

Implementations:

All the functions that are dealing with the organization of the data are found in the Python file utils.py The main idea of handling this data set:

- the Winner / Loser columns were changed in P1/P2 columns;
- a P1_won column was created with pseudo random values chosen from 1 and 0 with a rate of 50%, so in this way the P1_won column will not contain all the time the winner;
- the data frame was parsed and every time the P1_won was 0 and the P1 and P2 were switched, so the data stays correct;

Missing values: All the NaN values were replaced with the mean() value.

Factorization: Each non-numerical feature from the data set was factorized using the pandas library function: factorize().

Train/test data set: The training and test set were chosen with the sklearn function: *train_test_split*() and the test size was set at the value 0.25 of the training test.

Data Handling

After applying all functions and operations to the data set,this is the resulting data to be worked with further:

	Date	Tournament	Court	Surface	Round	Best of	Series	P1_won	P1	P2	P1Rank	P2Rank	P1Pts	P2Pts	AvgP2	AvgP1	P1_Experince	P2_Experince	P1_W/L	P2_W/L
0	2015-01-05	0	0	0	0	3	0	False	Simon G.	Duckworth J.	21.0	125.0	1730.0	430.0	1.20	4.31	0	0	100.0	100.0
1	2015-01-05	0	0	0	0	3	0	True	Kokkinakis T.	Benneteau J.	149.0	25.0	341.0	1365.0	1.47	2.62	0	0	100.0	100.0
2	2015-01-05	0	0	0	0	3	0	True	Chardy J.	Golubev A.	31.0	72.0	1195.0	691.0	3.30	1.32	0	0	100.0	100.0
3	2015-01-05	0	0	0	0	3	0	False	Querrey S.	Tomic B.	35.0	53.0	1090.0	797.0	2.25	1.61	0	0	100.0	100.0
4	2015-01-06	0	0	0	0	3	0	True	Kukushkin M.	Copil M.	69.0	201.0	705.0	242.0	2.53	1.50	0	0	100.0	100.0

Random Forest Classifier

About:

- Random forest is a supervised learning algorithm;
- used for both classification as well as regression (but it is mainly used for classification problems);
- as we know that a forest is made up of trees and more trees means more robust forest,

  similarly random forest algorithm creates decision trees on data samples and then gets the prediction from each of them based on a criterion and finally selects the best solution by means of voting;
- it is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Random Forest Classifier

About:

- Random forest is a supervised learning algorithm;
- used for both classification as well as regression (but it is mainly used for classification problems);
- as we know that a forest is made up of trees and more trees means more robust forest,

  similarly random forest algorithm creates decision trees on data samples and then gets the prediction from each of them based on a criterion and finally selects the best solution by means of voting;
- it is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Criterions: a comparison was made regarding the two possible criterions used by the random forest classifier:

- Gini impurity criterion: fit with a score of approx 0.82 on the test set
- Information gain criterion: fit with approx 0.81 on the test set

Random Forest Classifier

About:

- Random forest is a supervised learning algorithm;
- used for both classification as well as regression (but it is mainly used for classification problems);
- as we know that a forest is made up of trees and more trees means more robust forest,

  similarly random forest algorithm creates decision trees on data samples and then gets the prediction from each of them based on a criterion and finally selects the best solution by means of voting;
- it is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Criterions: a comparison was made regarding the two possible criterions used by the random forest classifier:

- Gini impurity criterion: fit with a score of approx 0.82 on the test set
- Information gain criterion: fit with approx 0.81 on the test set

Therefore, the Gini criterion will be further used because gave a slightly higher score

Random Forest Classifier

```
The score for the RandomForestClassifier with gini: 0.8126899696048632
The score for the RandomForestClassifier with entropy: 0.8119300911854104


gini index seems to fit the best:
0.8126899696048632
```

In [14]:
```
# Draw a little table to see the actual results
preds = forest.predict(X_test)
pd.crosstab(y_test, preds, rownames=['Actual Wins'],colnames=['Predicted Wins'])
```

Out[14]:

Predicted Wins	False	True
Actual Wins		
False	1058	229
True	264	1081

Logistical Regression Classifier

About:

- Logistic Regression is a technique borrowed by machine learning from the field of statistics;
- in machine learning it's a supervised learning classification algorithm used to predict the probability of a target variable;
- the target/dependent variable is binary in nature and coded as either 1(success/yes/win) or 0(failure/no/loss);

Logistical Regression Classifier

About:

- Logistic Regression is a technique borrowed by machine learning from the field of statistics;
- in machine learning it's a supervised learning classification algorithm used to predict the probability of a target variable;
- the target/dependent variable is binary in nature and coded as either 1(success/yes/win) or 0(failure/no/loss);

Customization: In regards with this model only one parameter was set custom: max_iter, because the model was reaching the maximum limit on a data frame this big.

Results: Logistical Regression classifier gave a score of only 0.65 on the test set.

Logistical Regression Classifier

In [16]:
```
regr.score(X_test, y_test)
```

Out[16]:
```
0.6546352583586627
```

In [17]:
```
preds = regr.predict(X_test)
pd.crosstab(y_test, preds, rownames=['Actual Wins'],colnames=['Predicted Wins'])
```

Out[17]:

Actual Wins \ Predicted Wins	False	True
False	827	460
True	449	896

Models on Australia Open 2019

It was noticed that the random forest classifier had a small boost in score of almost 0.02 (using the Gini criterion) and the logistical regression had a boost of 0.05, therefore still being slower than the RandomForestClassifier.

In [22]:
```
# We have the RandomForestClassifier and the LogisticRegression
score_list = [regr.score(test_df[features], test_df['P1_won']),forest.score(test_df[features], test_df['P1_won'])]

pd.DataFrame.from_dict(
    {
        "Name":["LogisticalRegression","RandomForestClassifier"],
        "Score":score_list
    }
)
```

Out[22]:

	Name	Score
0	LogisticalRegression	0.700787
1	RandomForestClassifier	0.834646

The choice is RandomForestClassifier,having the best score