|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | Информатика, искусственный интеллект и системы управления |
| КАФЕДРА | Системы обработки информации и управления |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

|  |
| --- |
| Использование методов машинного |
| обучения для прогнозирования |
| результатов футбольных матчей |
|  |
|  |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ5-33М |  |  |  | Н.В. Бакланов |
|  | (Группа) |  | (Подпись, дата) |  | (И.О.Фамилия) |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | Ю.Е. Гапанюк |
|  | (Подпись, дата) |  | (И.О.Фамилия) |

|  |  |  |  |
| --- | --- | --- | --- |
| Консультант |  |  |  |
|  | (Подпись, дата) |  | (И.О.Фамилия) |

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

|  |  |
| --- | --- |
| по теме | Использование методов машинного обучения для прогнозирования результатов |
| футбольных матчей | |

|  |  |
| --- | --- |
| Студент группы | ИУ5-33М |

|  |
| --- |
| Бакланов Никита Владимирович |

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

|  |
| --- |
| исследовательская |

|  |  |
| --- | --- |
| Источник тематики (кафедра, предприятие, НИР) | кафедра |

График выполнения НИР: 25% к 4 нед., 50% к 8 нед., 75% к 12 нед., 100% к 17 нед.

|  |  |
| --- | --- |
| ***Техническое задание*** |  |
|  | |
|  | |
|  | |

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \*\* листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

|  |
| --- |
|  |
|  |
|  |

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

|  |  |  |  |
| --- | --- | --- | --- |
| **Руководитель НИР** |  |  | Ю.Е. Гапанюк |
|  | (Подпись, дата) |  | (И.О.Фамилия) |
| **Студент** |  |  | Н.В. Бакланов |
|  | (Подпись, дата) |  | (И.О.Фамилия) |

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

стр.

[Введение 4](#_Toc121879007)

[Данные 4](#_Toc121879008)

[Обучение модели 8](#_Toc121879009)

[Результаты 9](#_Toc121879010)

[Заключение 11](#_Toc121879011)

[Список использованных источников 12](#_Toc121879012)

# **Введение**

Ставки на спорт - одна из идеальных задач для алгоритмов машинного обучения, особенно классификация нейронных сетей. Тонны доступных данных и четкая цель выбора победителя. Никогда нельзя будет делать 100% прогнозы на футбол (и вообще на спорт). Здесь решает человеческий фактор. Защитник поскользнулся и упустил нападающего соперника, который в итоге забил гол, в глаза вратаря светили лазером, и он не смог среагировать, нападающий травмировался уже на первых минутах матча и тд. Однако для работы букмекеров необходим расчет коэффициентов, поэтому данная модель может использоваться букмекерами для выбора коэффициентов на команды перед матчем.

Модель машинного обучения на Python c использованием библиотеки Scikit-learn, для прогнозирования результатов футбольных матчей Российской Премьер Лиги (РПЛ). Модель машинного обучения будет тренироваться на статистике матчей Российской Премьер Лиги (РПЛ) начиная с сезона 2015/2016, чтобы предсказывать результаты предстоящих игр. Данные взяты с сайта футбольной статистики wyscout.com.

# **Данные**

Подключаем необходимые библиотеки:

**import** pandas **as** pd

**import** numpy **as** np

**import** collections

Данные с матчами:

data = pd.read\_csv("RPL.csv", encoding = 'cp1251', delimiter=';')

data.head()



xG (expected goals) – это модель ожидаемых голов. В основе её лежит показатель ударов по воротам, на основе которого мы можем оценить сколько реально голов должна была забить команда, если учесть все удары которые она нанесла.

PPDA (Passes Allowed Per Defensive Action) — футбольный статистический показатель, который позволяет определить интенсивность прессинга в матче. Чем меньше значение PPDA, тем выше интенсивность игры в обороне.

PPDA = число передач, которое сделала атакующая команда / число действий в обороне

Мы будем прогнозировать результаты матчей для второй части сезона 2018/2019 (т.е. матчи, сыгранные в 2019 году). Список команд, играющих в этом сезоне (не учитывая Арсенал, Оренбург, Динамо, Крылья Советов и Енисей, т.к. у них либо отсутствует статистика за прошлые сезоны, либо статистики по ним мало):

RPL\_2018\_2019 = pd.read\_csv('Team Name 2018 2019.csv', encoding = 'cp1251')

teamList = RPL\_2018\_2019['Team Name'].tolist()

teamList

image

Удаляем матчи с командами, которые не участвуют в сезоне 2018/2019:

deleteTeam = [x **for** x **in** pd.unique(data['Команда']) **if** x **not** **in** teamList]

**for** name **in** deleteTeam:

data = data[data['Команда'] != name]

data = data[data['Соперник'] != name]

data = data.reset\_index(drop=True)

Функция, возвращающая статистику команды за сезон:

**def** **GetSeasonTeamStat**(team, season):

goalScored = 0 #Голов забито

goalAllowed = 0 #Голов пропущено

gameWin = 0 #Выиграно

gameDraw = 0 #Ничья

gameLost = 0 #Проиграно

totalScore = 0 #Количество набранных очков

matches = 0 #Количество сыгранных матчей

xG = 0 #Ожидаемые голы

shot = 0 #Удары

shotOnTarget = 0 #Удары в створ

cross = 0 #Навесы

accurateCross = 0 #Точные навесы

totalHandle = 0 #Владение мячом

averageHandle = 0 #Среднее владение мячом за матч

Pass = 0 #Пасы

accuratePass = 0 #Точные пасы

PPDA = 0 #Интенсивность прессинга в матче

**for** i **in** range(len(data)):

**if** (((data['Год'][i] == season) **and** (data['Команда'][i] == team) **and** (data['Часть'][i] == 2)) **or** ((data['Год'][i] == season-1) **and** (data['Команда'][i] == team) **and** (data['Часть'][i] == 1))):

matches += 1

goalScored += data['Забито'][i]

goalAllowed += data['Пропущено'][i]

**if** (data['Забито'][i] > data['Пропущено'][i]):

totalScore += 3

gameWin += 1

**elif** (data['Забито'][i] < data['Пропущено'][i]):

gameLost +=1

**else**:

totalScore += 1

gameDraw += 1

xG += data['xG'][i]

shot += data['Удары'][i]

shotOnTarget += data['Удары в створ'][i]

Pass += data['Передачи'][i]

accuratePass += data['Точные передачи'][i]

totalHandle += data['Владение'][i]

cross += data['Навесы'][i]

accurateCross += data['Точные навесы'][i]

PPDA += data['PPDA'][i]

averageHandle = round(totalHandle/matches, 3) #Владение мячом в среднем за матч

**return** [gameWin, gameDraw, gameLost,

goalScored, goalAllowed, totalScore,

round(xG, 3), round(PPDA, 3),

shot, shotOnTarget,

Pass, accuratePass,

cross, accurateCross,

round(averageHandle, 3)]

Пример использования функции:

GetSeasonTeamStat("Спартак", 2018) #Статистика Спартака за сезон 2017/2018

image  
  
Для удобства можем дописать код:

returnNames = ["Выиграно", "Ничья", "Проиграно",

"\nГолов забито", "Голов пропущено", "\nНабрано очков",

"\nxG (за сезон)", "PPDA (за сезон)",

"\nУдары", "Удары в створ",

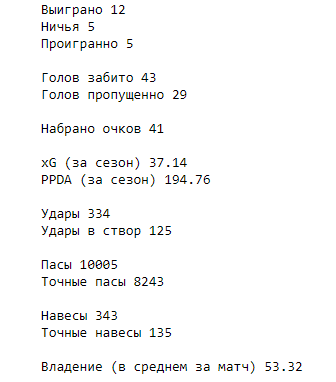
"\nПасы", "Точные пасы",

"\nНавесы", "Точные навесы",

"\nВладение (в среднем за матч)"]

**for** i, n **in** zip(returnNames, GetSeasonTeamStat("Спартак", 2018)):

print(i, n)



Функция, которая будет возвращать статистику всех команд за сезон:

**def** **GetSeasonAllTeamStat**(season):

annual = collections.defaultdict(list)

**for** team **in** teamList:

team\_vector = GetSeasonTeamStat(team, season)

annual[team] = team\_vector

**return** annual

# **Обучение модели**

Напишем функцию, которая будет возвращать обучающие данные. Она создает словарь с векторами команд за все сезоны. Для каждой игры функция рассчитывает разницу между векторами команд за определенный сезон и записывает в xTrain. Затем функция присваивает yTrain значение 1, если команда хозяев выигрывает, и 0 в противном случае.

**def** **GetTrainingData**(seasons):

totalNumGames = 0

**for** season **in** seasons:

annual = data[data['Год'] == season]

totalNumGames += len(annual.index)

numFeatures = len(GetSeasonTeamStat('Зенит', 2016)) #случайная команда для определения размерности

xTrain = np.zeros(( totalNumGames, numFeatures))

yTrain = np.zeros(( totalNumGames ))

indexCounter = 0

**for** season **in** seasons:

team\_vectors = GetSeasonAllTeamStat(season)

annual = data[data['Год'] == season]

numGamesInYear = len(annual.index)

xTrainAnnual = np.zeros(( numGamesInYear, numFeatures))

yTrainAnnual = np.zeros(( numGamesInYear ))

counter = 0

**for** index, row **in** annual.iterrows():

team = row['Команда']

t\_vector = team\_vectors[team]

rivals = row['Соперник']

r\_vector = team\_vectors[rivals]

diff = [a - b **for** a, b **in** zip(t\_vector, r\_vector)]

**if** len(diff) != 0:

xTrainAnnual[counter] = diff

**if** team == row['Победитель']:

yTrainAnnual[counter] = 1

**else**:

yTrainAnnual[counter] = 0

counter += 1

xTrain[indexCounter:numGamesInYear+indexCounter] = xTrainAnnual

yTrain[indexCounter:numGamesInYear+indexCounter] = yTrainAnnual

indexCounter += numGamesInYear

**return** xTrain, yTrain

Поучаем обучающие данные за все сезоны с 2015/2016 по 2018/2019.

years = range(2016,2019)

xTrain, yTrain = GetTrainingData(years)

Для прогнозирования вероятности выигрыша будем использовать алгоритм машинного обучения LinearRegression из библиотеки Scikit-Learn.

**from** sklearn.linear\_model **import** LinearRegression

model = LinearRegression()

model.fit(xTrain, yTrain)

Напишем функцию, которая будет возвращать прогнозы. Она будет возвращать значение в промежутке от 0 до 1, где 0 — это проигрыш, а 1 — это выигрыш.

**def** **createGamePrediction**(team1\_vector, team2\_vector):

diff = [[a - b **for** a, b **in** zip(team1\_vector, team2\_vector)]]

predictions = model.predict(diff)

**return** predictions

# **Результаты**

Для примера посмотрим прогнозы алгоритма на матч Зенит — Спартак

team1\_name = "Зенит"

team2\_name = "Спартак"

team1\_vector = GetSeasonTeamStat(team1\_name, 2019)

team2\_vector = GetSeasonTeamStat(team2\_name, 2019)

print ('Вероятность, что выиграет ' + team1\_name + ':', createGamePrediction(team1\_vector, team2\_vector))

print ('Вероятность, что выиграет ' + team2\_name + ':', createGamePrediction(team2\_vector, team1\_vector))

image  
  
Получается, что в матче Зенит — Спартак вероятность победы Зенита составляет 47% (17.03.2019 Спартак 1-1 Зенит).

Делаем прогноз учитывая следующее:  
До 40% — команда точно не выиграет (проигрыш или ничья)  
От 40% до 60% — высокая вероятность ничьи  
От 60% — команда точно не проиграет (победа или ничья)  
  
Выведем прогнозы для ЦСКА против всех остальных клубов

**for** team\_name **in** teamList:

team1\_name = "ЦСКА"

team2\_name = team\_name

**if**(team1\_name != team2\_name):

team1\_vector = GetSeasonTeamStat(team1\_name, 2019)

team2\_vector = GetSeasonTeamStat(team2\_name, 2019)

print(team1\_name, createGamePrediction(team1\_vector, team2\_vector), " - ", team2\_name, createGamePrediction(team2\_vector, team1\_vector,))

  
Алгоритм дал верный прогноз почти на все матчи, которые не закончились в ничью. Единственный неточный прогноз: ЦСКА — Зенит. Вероятность победы ЦСКА выше на 0.001, можно было предположить, что команды равны по силе и сыграют в ничью, но в итоге победил Зенит (3-1).

# **Заключение**

Алгоритм дал верный прогноз почти на все матчи, которые не закончились в ничью. Однако, он учитывает лишь статистику матчей (и то только 15 основных параметров), а результат в футболе зависит от многих факторов. Даже состояние поля или погода могут повлиять на результат игры. Дальше хотелось бы увеличить количество признаков, создать тестовую выборку, попробовать различные алгоритмы, настроить модель и получить максимально точные прогнозы.

# **Список использованных источников**

1. https://towardsdatascience.com/machine-learning-for-sports-betting-not-a-basic-classification-problem-b42ae4900782.
2. <https://mariamsulakian.com/2018/02/01/machine-learning-predicting-the-2018-epl-matches/>.