МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южный федеральный университет»
Институт высоких технологий и пьезотехники

Кафедра прикладной информатики и инноватики

Направление: 09.03.03 "Прикладная информатика"

Большие данные

Отчёт по проекту

«Анализ футбольного рынка»

Выполнили студенты 3 курса 7 группы

_____ Колисниченко М.В.

подпись

_____ Сылкин А.А.

подпись

Проверил старший преподаватель

_____ Яценко Д.В.

подпись

Ростов-на-Дону, 2024

СОДЕРЖАНИЕ

1. Описание проблемы

Существует тенденция, что в разных национальных чемпионатах для того, чтобы у игрока поднялась стоимость на m миллионов евро, ему нужно сделать разное количество полезных действий. Другими словами, 1 полезное действие в зависимости от лиги, где играет футболист, имеет разную стоимость.

Например, Английская Премьер-Лига (АПЛ) считается одной из богатых лиг, и бытует мнение, что ценники на футболистов в чемпионате совсем не соизмеримые с тем, что делает игрок на поле.

2. Постановка задачи

При помощи полученных во время учёбы знаний, а также дополнительного материала, изученного при использовании различной литературы, создать решение, по которому можно проанализировать, как меняется стоимость футболистов от различных факторов, и решить тем самым следующие подзадачи:

- выяснить какие данные нужны и полезны;
- преобразовать данные к необходимому виду;
- обучить линейную регрессию, дав на вход преобразованные данные;
- получить коэффициенты регрессии нужных признаков;
- визуализировать результаты.

Поскольку выбранный набор данных имеет полезные действия атакующего характера – голы и голевые передачи, было решено рассмотреть случаи роста цен у нападающих и полузащитников.

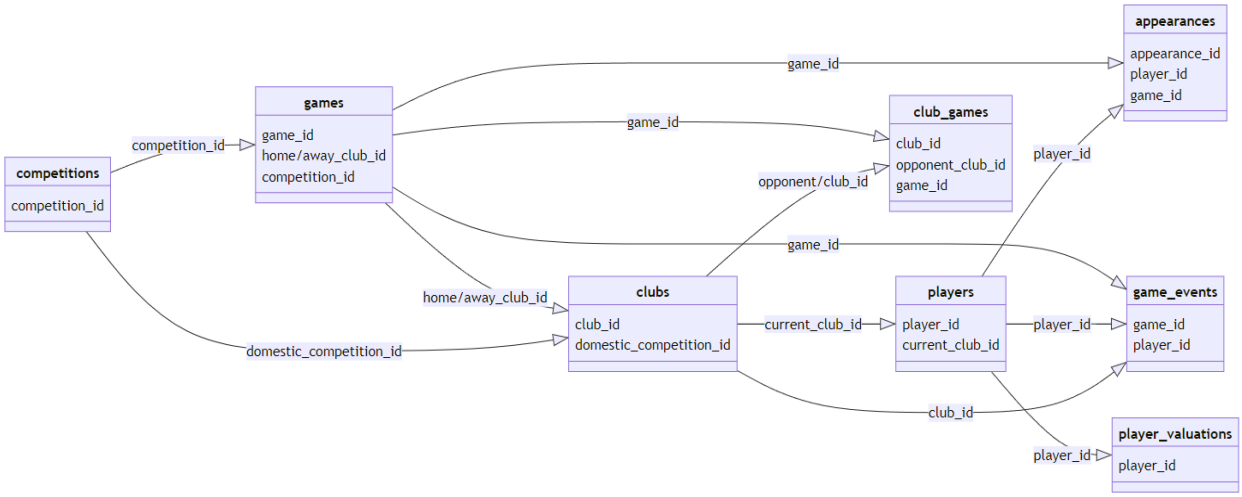
Для объективности данных было сделано решение рассмотреть ценообразования игроков с сезона 2017/2018. В этом сезоне нападающий «Барселоны» Неймар перешёл во французский «ПСЖ» за рекордные в наши дни 222 млн. евро, после чего цены на рынке существенно изменились.

Рассмотренные футбольные лиги стали – классический топ-5 (Англия, Испания, Италия, Германия, Франция) и Россия с Турцией.

Было решено использовать учебный кластер и PySpark для обработки данных.

3. Описание набора данных

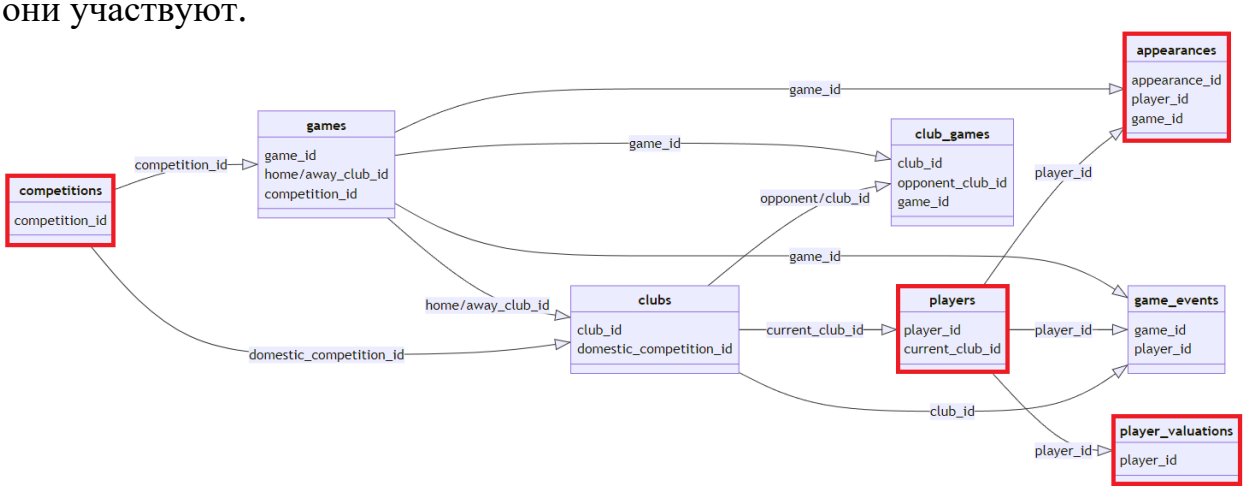
Набор данных **Football Data from Transfermarket** состоит из нескольких CSV-файлов с информацией о соревнованиях, играх, клубах, игроках и выступлениях, которая автоматически обновляется раз в неделю. Каждый файл содержит атрибуты сущности и идентификаторы, которые можно использовать для их объединения.



4. Ход работы

4.1. Предобработка данных

1. Исходя из поставленной задачи было выяснено, что наиболее нужная и полезная информация хранится в CSV-файлах об игроках, их стоимости в разные периоды времени, их появления на поле и соревнования, в которых они участвуют.



При помощи Spark Dataframe API выгрузили соответствующие файлы и назначили схему с интересующими нас признаками (столбцами).

1) players:

```
In [6]: playersDF = spark.read.csv("BDProject/SliceDataset/players", header=True, schema=schemaPlayers)

In [7]: playersDF.printSchema()

root
 |-- player_id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- country_of_citizenship: string (nullable = true)
 |-- date_of_birth: date (nullable = true)
 |-- position: string (nullable = true)
 |-- sub_position: string (nullable = true)
```

2) player_valuations:

```
In [12]: playerValuationsDF = spark.read.csv("BDProject/SliceDataset/playerValuations", header=True
                                             schema=schemaPlayerValuations)

In [13]: playerValuationsDF.printSchema()

root
 |-- player_id: integer (nullable = true)
 |-- date: date (nullable = true)
 |-- market_value_in_eur: integer (nullable = true)
```

3) competitions:

```
In [33]: competitionsDF.printSchema()

root
 |-- competition_id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- sub_type: string (nullable = true)
 |-- type: string (nullable = true)
 |-- country_id: integer (nullable = true)
 |-- country_name: string (nullable = true)
 |-- domestic_league_code: string (nullable = true)
```

4) appearances:

```
In [19]: appearancesDF = spark.read.csv("BDProject/SliceDataset/appearances", header=True
                                         schema=schemaAppearances)

In [20]: appearancesDF.printSchema()

root
 |-- player_id: integer (nullable = true)
 |-- p_club_id: integer (nullable = true)
 |-- date: date (nullable = true)
 |-- player_name: string (nullable = true)
 |-- competition_id: string (nullable = true)
 |-- yellow_cards: integer (nullable = true)
 |-- red_cards: integer (nullable = true)
 |-- goals: integer (nullable = true)
 |-- assists: integer (nullable = true)
 |-- minutes_played: integer (nullable = true)
```

2. Работа с playerValuationsDF

Преобразовали playerValuationsDF таким образом, чтобы он отображал информацию о начальной стоимости и конечной стоимости в конкретном сезоне.

2017/2018

In [36]:
```
start_date = '2017-07-01'
last_date = '2018-06-13'
seasonPlayerValuations = playerValuationsDF.where("date >= '2017-07-01' and date <= '2018-06-13'")
seasonPlayerValuations.show(20)
```

```
+---------+----------+------------------+
|player_id|      date|market_value_in_eur|
+---------+----------+------------------+
|      992|2017-07-01|            100000|
|     6033|2017-07-01|            400000|
|     6824|2017-07-01|             50000|
|    13073|2017-07-01|            200000|
|    15570|2017-07-01|          11000000|
```

Start valuations

In [35]:
```
from pyspark.sql.functions import *

startPlayerDates = seasonPlayerValuations.groupBy("player_id").agg(min("date").alias("date"))
startPlayerDates.show()
```

```
[Stage 15:=======================================>                 (2 + 1) / 3]
```

```
+---------+----------+
|player_id|      date|
+---------+----------+
|   243856|2017-07-12|
|    85349|2017-07-14|
|   222556|2017-07-14|
|   382774|2017-07-14|
```

In [36]:
```
startPlayerValuations = startPlayerDates.join(seasonPlayerValuations, ["player_id", "date"])
```

In [37]:
```
startPlayerValuations = startPlayerValuations.select("player_id",
                                                     startPlayerValuations["date"].alias("start_date"),
                                                startPlayerValuations["market_value_in_eur"].alias("start_val"))
```

In [38]:
```
startPlayerValuations.show()
```

```
+---------+----------+---------+
|player_id|start_date|start_val|
+---------+----------+---------+
|      992|2017-07-01|   100000|
|     6033|2017-07-01|   400000|
|     6824|2017-07-01|    50000|
|    13073|2017-07-01|   200000|
```

Last valuations

In [40]:
```
lastPlayerDates = seasonPlayerValuations.groupBy("player_id").agg(max("date").alias("date"))
lastPlayerDates.orderBy(col("date").desc()).show()
```

```
[Stage 23:================================================>      (175 + 4) / 200]
```

```
+---------+----------+
|player_id|      date|
+---------+----------+
|   149928|2018-06-13|
|   148656|2018-06-13|
|    59823|2018-06-13|
|   131090|2018-06-13|
```

```
In [42]: lastPlayerValuations = lastPlayerDates.join(seasonPlayerValuations, ["player_id", "date"])

In [43]: lastPlayerValuations = lastPlayerValuations.select("player_id",
                                                            lastPlayerValuations["date"].alias("last_date"),
                                                            lastPlayerValuations["market_value_in_eur"].alias("last_val"))

In [44]: lastPlayerValuations.orderBy(col("date").desc()).show()
```

```
+---------+----------+--------+
|player_id| last_date|last_val|
+---------+----------+--------+
|     3979|2018-06-13| 1000000|
|    24956|2018-06-13|  100000|
|     5404|2018-06-13| 2000000|
|     9819|2018-06-13|  250000|
```

Вычислили разницу между конечной и стартовой стоимостью в сезоне, тем самым получив рост (положительный или отрицательный) в цене за игровой год.

Outcome valuations and difference between them for the season

```
In [47]: valsDF = startPlayerValuations.join(lastPlayerValuations, "player_id")

In [49]: diffVal = valsDF.last_val - valsDF.start_val

In [50]: valsDF = valsDF.withColumn("change", diffVal)

In [51]: valsDF.orderBy(col("change").desc()).show(10)
```

```
[Stage 53:====================>
```

```
+---------+----------+---------+----------+---------+---------+
|player_id|start_date|start_val| last_date| last_val|   change|
+---------+----------+---------+----------+---------+---------+
|   148455|2017-10-23| 40000000|2018-05-28|150000000|110000000|
|    88755|2017-10-23| 75000000|2018-05-28|150000000| 75000000|
|   132098|2017-10-23| 80000000|2018-05-28|150000000| 70000000|
|   266302|2017-10-13| 30000000|2018-06-07| 90000000| 60000000|
|   105470|2017-10-13| 10000000|2018-06-07| 60000000| 50000000|
|   296622|2017-08-17| 30000000|2018-05-30| 75000000| 45000000|
|   192565|2017-10-23| 45000000|2018-05-28| 90000000| 45000000|
|   134425|2017-10-23| 50000000|2018-05-28| 90000000| 40000000|
|   206050|2017-10-13| 70000000|2018-06-07|110000000| 40000000|
|   207929|2017-10-23| 60000000|2018-05-28|100000000| 40000000|
+---------+----------+---------+----------+---------+---------+
```

3. Работа с competitionsDF

Выяснили, какие competition_id имеют интересующие нас лиги.
England – «GB1», Spain – «ES1», Italy – «IT1», Germany – «L1», France – «FR1», Russia – «RU1», Turkey – «TR1».

```
In [61]: competitionsDF.where("country_name in ('England', 'Spain', 'Italy', 'Germany', 'France', 'Russia', 'Turkey')") \
             .orderBy("country_name").show()
```

```
+--------------+--------------------+-----------------+--------------+----------+------------+---+
|competition_id|                name|         sub_type|          type|country_id|country_name|dlc|
+--------------+--------------------+-----------------+--------------+----------+------------+---+
|           CGB|             efl-cup|       league_cup|         other|       189|     England|GB1|
|           FAC|              fa-cup|      domestic_cup|  domestic_cup|       189|     England|GB1|
|          GBCS|    community-shield|domestic_super_cup|         other|       189|     England|GB1|
|           GB1|      premier-league|       first_tier|domestic_league|       189|     England|GB1|
|           FR1|             ligue-1|       first_tier|domestic_league|        50|      France|FR1|
|          FRCH|trophee-des-champ...|domestic_super_cup|         other|        50|      France|FR1|
|           L1|           bundesliga|       first_tier|domestic_league|        40|     Germany| L1|
|           DFL|        dfl-supercup|domestic_super_cup|         other|        40|     Germany| L1|
|           DFB|           dfb-pokal|      domestic_cup|  domestic_cup|        40|     Germany| L1|
|           SCI| supercoppa-italiana|domestic_super_cup|         other|        75|       Italy|IT1|
|           IT1|             serie-a|       first_tier|domestic_league|        75|       Italy|IT1|
|           CIT|          italy-cup|      domestic_cup|  domestic_cup|        75|       Italy|IT1|
|           RUP|         russian-cup|      domestic_cup|  domestic_cup|       141|      Russia|RU1|
|           RU1|       premier-liga|       first_tier|domestic_league|       141|      Russia|RU1|
|          RUSS|   russian-super-cup|domestic_super_cup|         other|       141|      Russia|RU1|
|           SUC|          supercopa|domestic_super_cup|         other|       157|       Spain|ES1|
|           ES1|             laliga|       first_tier|domestic_league|       157|       Spain|ES1|
|           CDR|        copa-del-rey|      domestic_cup|  domestic_cup|       157|       Spain|ES1|
|           TR1|          super-lig|       first_tier|domestic_league|       174|      Turkey|TR1|
+--------------+--------------------+-----------------+--------------+----------+------------+---+
```

4. Работа с appearancesDF

Преобразовали appearancesDF в соответствии с сезоном и лигой.
cid – competition_id, который был найден выше, date – дата, которая фиксирует результативное действие игрока.

Conditions - League & season

```
In [55]: filtered_games = appearancesDF.filter(
             (col("cid") == "GB1") &
             (col("date") >= start_date) &
             (col("date") <= last_date)
         )
```

Сделали группировку по player_id. Таким образом выяснили, сколько результативных действий делает игрок за сезон.
Goals – голы, assists – голевые передачи, yc – жёлтые карточки, rc – красные карточки, mins – среднее количество минут, sum_mins – суммарное количество минут.

```
In [82]: allFeaturesDF = filtered_games \
             .groupBy("player_id") \
             .agg(sum("goals").alias("goals"), sum("assists").alias("assists"), sum("yc").alias("yc"),
                  sum("rc").alias("rc"), round(mean("mins"), 2).alias("mins"), sum("mins").alias("sum_mins"))
```

```
In [83]: allFeaturesDF.orderBy(col("goals").desc()).show(10)
```

```
[Stage 90:=====================================>                  (2 + 1) / 3]
```

```
+---------+-----+-------+---+---+-----+--------+
|player_id|goals|assists| yc| rc| mins|sum_mins|
+---------+-----+-------+---+---+-----+--------+
|   148455|   32|     11|  1|  0|81.11|    2920|
|   132098|   30|      3|  5|  0|83.32|    3083|
|    26399|   21|      6|  2|  0|78.76|    1969|
|   197838|   20|      1|  3|  0|88.05|    3258|
|   134425|   18|     15|  5|  0|78.61|    2594|
|    96341|   16|      7|  4|  0|84.38|    2869|
|   131789|   15|      7|  1|  0|75.08|    2778|
|    93720|   14|      5|  1|  0|69.09|    2211|
|   363205|   13|      4|  6|  0|57.66|    1672|
|    50202|   12|      4|  2|  0|71.56|    2433|
+---------+-----+-------+---+---+-----+--------+
only showing top 10 rows
```

5. Объединение двух результирующих DF в один

Result table of the season

```
In [86]: resultDF = valsDF.join(allFeaturesDF, "player_id")

In [87]: resultDF.show(10)
```

```
+---------+----------+---------+----------+---------+--------+-----+-------+---+---+-----+--------+
|player_id|start_date|start_val| last_date| last_val|  change|goals|assists| yc| rc| mins|sum_mins|
+---------+----------+---------+----------+---------+--------+-----+-------+---+---+-----+--------+
|   253765|2017-10-23| 12000000|2018-05-28| 15000000| 3000000|    0|      1|  7|  0| 90.0|    2520|
|    37526|2018-01-02|  8000000|2018-05-28|  7000000|-1000000|    0|      0|  5|  0|85.53|    1454|
|   129627|2018-01-02| 12000000|2018-05-28| 10000000|-2000000|    7|      0|  2|  0|42.92|    1030|
|    93905|2018-01-02|  5000000|2018-01-02|  5000000|       0|    0|      0|  0|  0| 23.0|      46|
|    44792|2017-10-23|  8000000|2018-05-28|  8000000|       0|    0|      1|  3|  0|85.76|    2830|
|    50202|2018-01-02|100000000|2018-05-28|110000000|10000000|   12|      4|  2|  0|71.56|    2433|
|   111524|2017-07-20|  1500000|2018-05-28|  5000000| 3500000|    2|      0|  4|  0|89.13|    3387|
|    85475|2018-01-02|  8000000|2018-05-28|  8000000|       0|    1|      1|  5|  0|89.23|    2677|
|   111114|2018-01-02|  1000000|2018-05-28|  2500000| 1500000|    1|      2|  1|  0| 90.0|    1440|
|   292417|2017-08-16| 10000000|2018-05-28| 20000000|10000000|    3|      2|  2|  0| 48.4|    1694|
+---------+----------+---------+----------+---------+--------+-----+-------+---+---+-----+--------+
only showing top 10 rows
```

6. Проверка правильности результатов

Объединили результирующий DF с playersDF для проверки реальности данных.

```
In [88]: resultDF.join(playersDF, "player_id").select("player_id", "name", "start_date", "start_val", "last_date",
                                                      "last_val", "goals", "assists", "mins", "change") \
         .orderBy(col("change").desc()).show(30, truncate=False)
```

```
[Stage 132:==============================>                    (114 + 4) / 200]
```

```
+---------+---------------------+----------+---------+----------+---------+-----+-------+-----+---------+
|player_id|name                 |start_date|start_val|last_date |last_val |goals|assists|mins |change   |
+---------+---------------------+----------+---------+----------+---------+-----+-------+-----+---------+
|148455   |Mohamed Salah        |2017-10-23|40000000 |2018-05-28|150000000|32   |11     |81.11|110000000|
|88755    |Kevin De Bruyne      |2017-10-23|75000000 |2018-05-28|150000000|8    |16     |83.38|75000000 |
|132098   |Harry Kane           |2017-10-23|80000000 |2018-05-28|150000000|30   |3      |83.32|70000000 |
|192565   |Leroy Sané           |2017-10-23|45000000 |2018-05-28|90000000 |10   |15     |75.75|45000000 |
|134425   |Raheem Sterling      |2017-10-23|50000000 |2018-05-28|90000000 |18   |15     |78.61|40000000 |
|207929   |Dele Alli            |2017-10-23|60000000 |2018-05-28|100000000|9    |11     |82.53|40000000 |
```

7. Работа с playersDF

Вычислили разницу между датой рождения игрока и датой сезона, тем самым получив возраст игрока на конкретный сезон.

```
In [75]: age_days = datediff(lit(last_date), col("date_of_birth"))

In [76]: age_days

Out[76]: Column<'datediff(2018-06-13, date_of_birth)'>

In [77]: age = round(age_days / 365, 2)
```

Использовали dropna, т.к. у некоторых игроков отсутствует дата рождения. Выбрали нужные столбцы: country_of_citizenship – паспорт, position – общая позиция игрока, age – возраст.

```
In [80]: slicedPlayersDF = playersDF.select("player_id", "name",
                                            playersDF["country_of_citizenship"].alias("citizenship"),
                                            playersDF["position"].alias("pos"), age.alias("age")).dropna(subset=["age"])
```

```
In [82]: slicedPlayersDF.show(truncate=False)
```

```
+---------+--------------------+--------------+----------+-----+
|player_id|name                |citizenship   |pos       |age  |
+---------+--------------------+--------------+----------+-----+
|10       |Miroslav Klose      |Germany       |Attack    |40.04|
|26       |Roman Weidenfeller  |Germany       |Goalkeeper|37.88|
|65       |Dimitar Berbatov    |Bulgaria      |Attack    |37.39|
|77       |Lúcio               |Brazil        |Defender  |40.13|
```

8. Было сделано предположение, что на стоимость также влияют полезные действия в еврокубках, потому **пункты 3 и 4** были повторены для Лиги Чемпионов и Лиги Европы (1-й и 2-й по значимости клубные международные турниры).
9. Был написан общий скрипт с циклами описанных выше действий для выбранных лиг, сезонов, возрастов и позиций.

```
dates = [("2017-07-01", "2018-06-20"), ("2018-07-01", "2019-06-20"),
         ("2019-07-01", "2020-06-20"), ("2020-07-01", "2021-06-20")
         ("2021-07-01", "2022-06-20"), ("2022-07-01", "2023-06-20"),
         ("2023-07-01", "2024-06-20")]

passport = [("GB1", "England"), ("ES1", "Spain"), ("IT1", "Italy"), ("L1", "Germany"), ("FR1", "France"),
            ("RU1", "Russia"), ("TR1", "Turkey")]

age_groups = [(16, 20), (21, 25), (26, 30)]

positions = ["Attack", "Midfield"]
```

В итоге у нас вышло 42 DF’a (7 стран * 3 возрастные группы * 2 позиции).

4.2. Получение значений

1. Создали список, содержащий интересующие названия признаков. Исключили «change», т.к. является меткой.

```
In [138]: columns = [fullDF.columns[6]] + fullDF.columns[9:]
          columns.remove("change")
```

```
In [139]: columns
```

```
Out[139]: ['start_val',
           'goals',
           'assists',
           'yc',
           'rc',
           'mins',
           'sum_mins',
           'cl_goals',
           'cl_assists',
           'cl_yc',
           'cl_rc',
           'cl_mins',
           'el_goals',
           'el_assists',
           'el_yc',
           'el_rc',
           'el_mins',
           'is_native']
```

2. Создали преобразователь, чтобы собрать столбцы объектов в виде Vector Column. Чтобы выполнить это воспользовались VectorAssembler().

```
In [140]: from pyspark.ml.feature import VectorAssembler

          assembler = VectorAssembler(inputCols=columns, outputCol="features")

          transformData = assembler.transform(fullDF)
```

3. Использовали класс LinearRegression для создания оценщика линейной регрессии.

```
In [142]: from pyspark.ml.regression import LinearRegression

          lr = LinearRegression(labelCol="change")
```

```
In [143]: modelLR = lr.fit(transformData)
```

4. Получили значения **коэффициентов** линейной регрессии.

```
coefficients = modelLR.coefficients

features = dict(zip(columns, coefficients))
```

```
{'is_native': -2619333.1155449697, 'start_val': -0.46416433263065937, 'total_goals': 1752564.5289292869, 'total_as
sists': 4436245.810714901, 'yellow_cards': -2022320.9263595508, 'red_cards': 0.0, 'minutes_played': -78362.2388162
451, 'cl_goals': 11229677.698893437, 'cl_assists': -5480886.739217845, 'cl_yc': 1822528.6500744359, 'cl_rc': 0.0,
'cl_mins': -1768.4591622910423, 'el_goals': 3133445.049813079, 'el_assists': 552764.1870681728, 'el_yc': 7485659.2
71240132, 'el_rc': 0.0, 'el_mins': 34873.06781798694}
```

Они определяют средние изменения результата с изменением соответствующего фактора на 1 при неизменном значении других факторов, зафиксированных на среднем уровне.

5. Оценили качество построенной модели с помощью коэффициента детерминации.

```
r2 = modelLR.summary.r2
print(f"Коэффициент детерминации (R²): {r2}")
```

```
Коэффициент детерминации (R²): 0.8663024586861284
```

Коэффициент детерминации оценивает, какая доля вариации зависимой переменной объясняется независимыми переменными модели. Значение R^2 близкое к 1 указывает на хорошее качество модели

4.3. Анализ и визуализация результатов

1. Написали общий скрипт описанных действий в **4.2.** для полученных ранее DF’в.

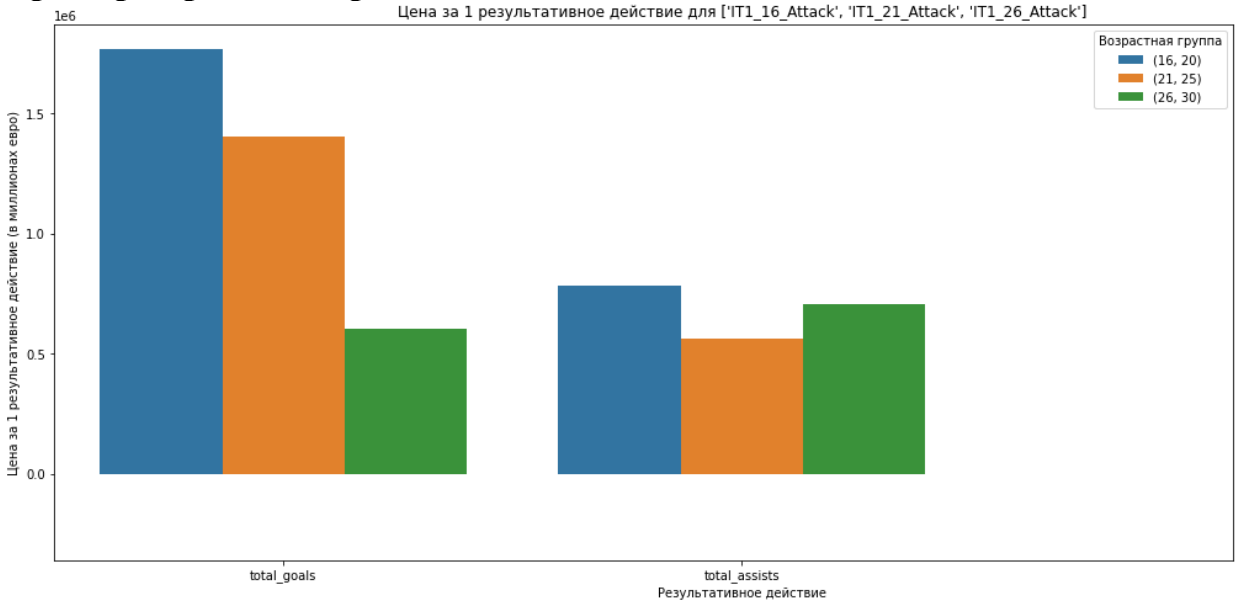
2. При помощи библиотеки питона matplotlib построили столбчатые диаграммы 3-х видов:

1) стоимость голов и голевых передач по позиции с разными возрастами в рамках одной лиги;

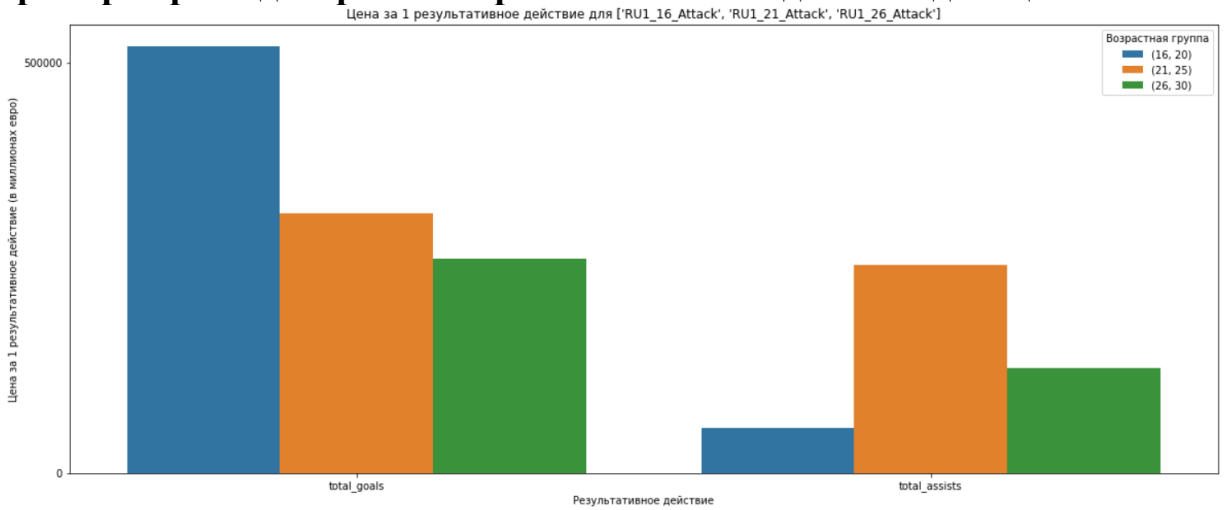
2) стоимость голов и голевых передач игроков разных позиций в рамках одной лиги;

3) стоимость голов и голевых передач игроков в разных лигах.

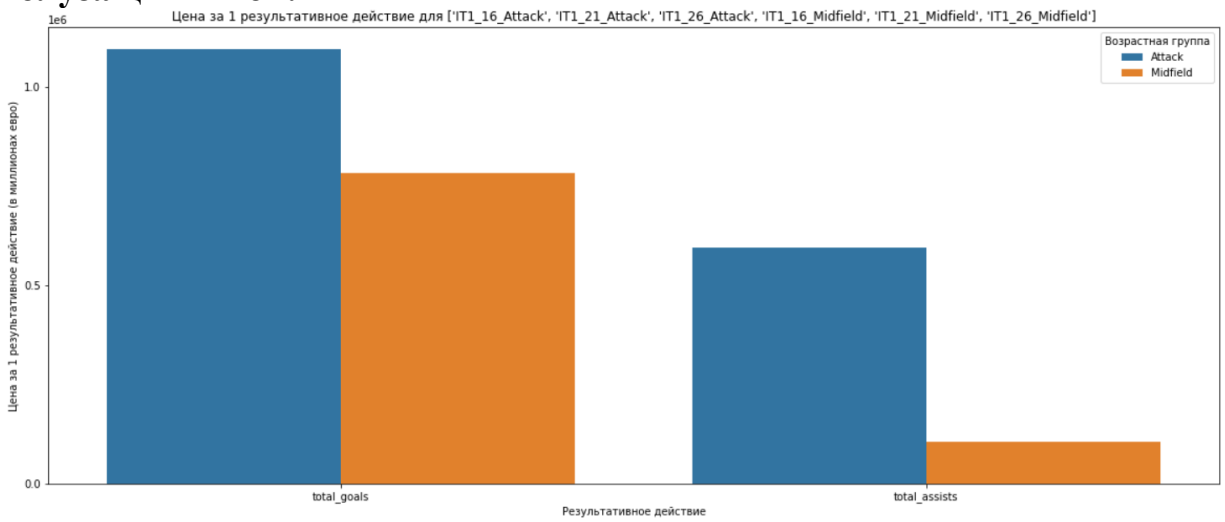
Пример первой диаграммы в итальянской лиге для нападающих:



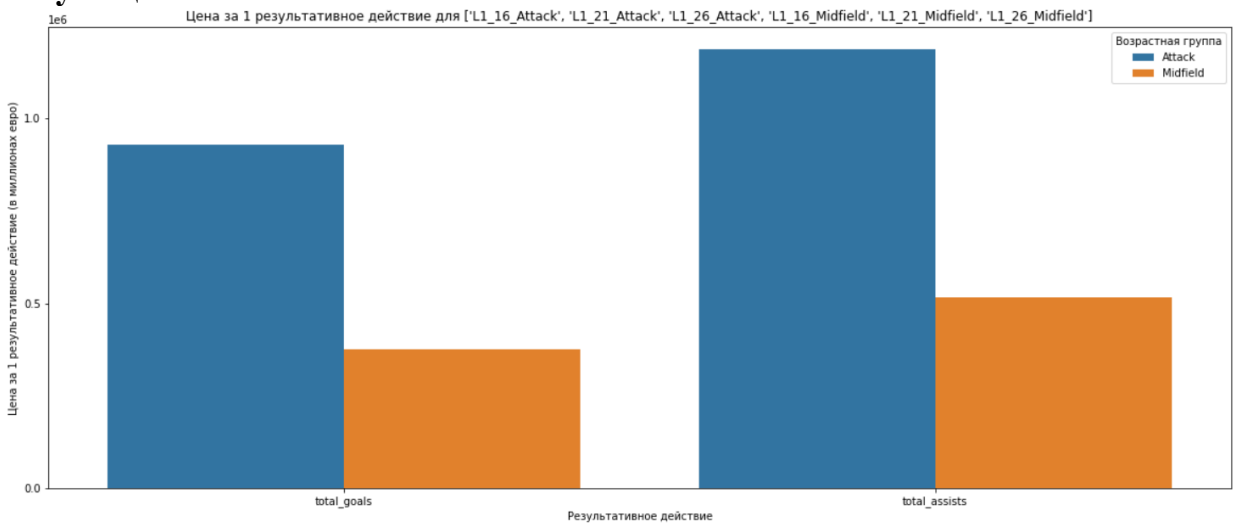
Пример первой диаграммы в российской лиге для нападающих:



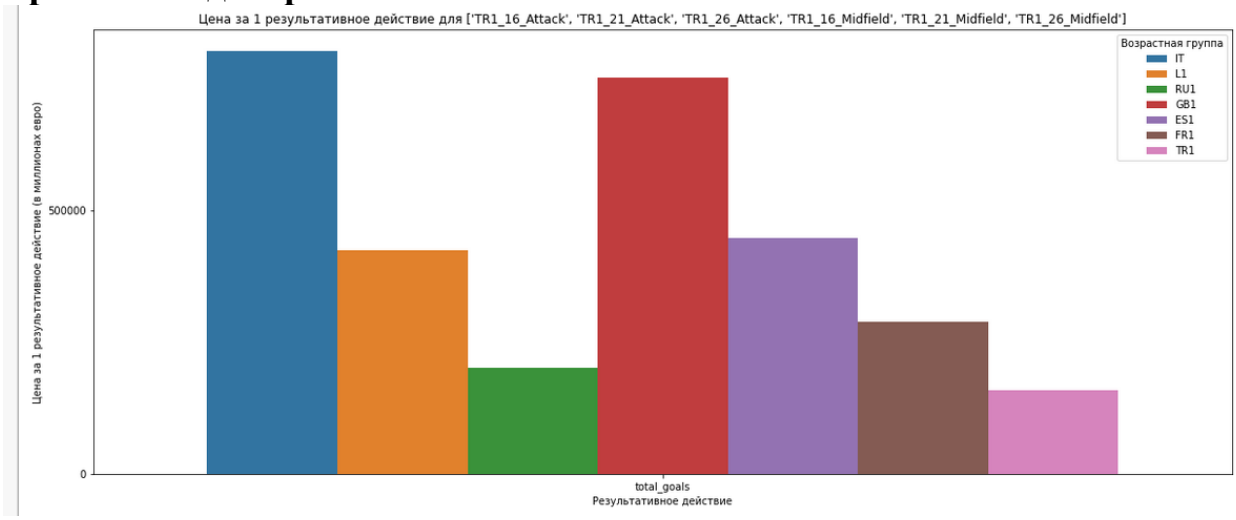
Пример второй диаграммы в итальянской лиге для нападающих и полузащитников:

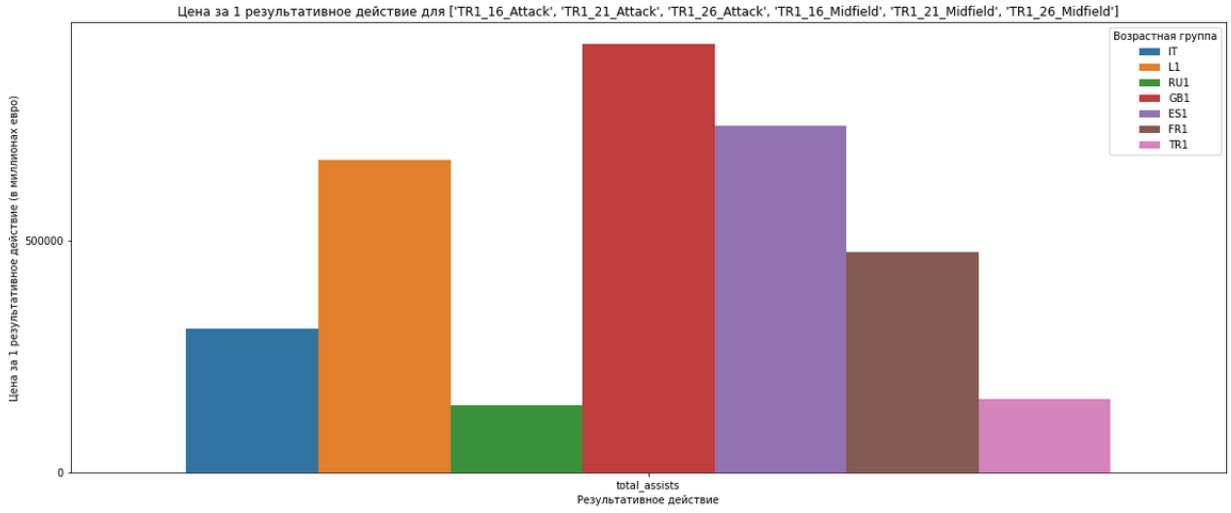


Пример второй диаграммы в немецкой лиге для нападающих и полузащитников:



Третий тип диаграммы:





5. Выводы

Исходя из результатов и соответствующим им графикам, можно убедиться в том, что главный тезис футбольных болельщиков частично верный – в АПЛ результативное действие стоит дороже, чем в других национальных чемпионатах. Однако, можно увидеть и то, что в итальянской Серии А также дорого ценятся голы.

Также можно сделать вывод о том, что возраст и позиция также влияет на рост цены: чем моложе футболист – тем дороже его гол/передача, и аналогично с позициями нападающего и полузащитника.