In [1]:
```
import findspark
findspark.init()
findspark.find()

import pyspark
from pyspark import SparkContext, SparkConf, SQLContext
from pyspark.sql import SparkSession

from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .master("local[*]") \
    .appName("GenericAppName") \
    .getOrCreate()
sc=spark.sparkContext
sqlContext = SQLContext(sc)
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/11/28 01:56:47 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/11/28 01:56:47 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/11/28 01:56:47 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/11/28 01:56:47 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
```

Task I

In [5]:
```
# Ingest data 2015-2022
from pyspark import SparkFiles

players_15 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
players_16 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
players_17 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
players_18 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
players_19 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
players_20 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
players_21 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
players_22 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_DATA/pla
```

In [6]:
```
print(players_15.columns[55])
```

```
movement_agility
```

In [7]:
```
#Add new column for the year
from pyspark.sql.functions import lit

players_15 = players_15.withColumn('Year', lit(2015))
players_16 = players_16.withColumn('Year', lit(2016))
players_17 = players_17.withColumn('Year', lit(2017))
players_18 = players_18.withColumn('Year', lit(2018))
players_19 = players_19.withColumn('Year', lit(2019))
players_20 = players_20.withColumn('Year', lit(2020))
players_21 = players_21.withColumn('Year', lit(2021))
players_22 = players_22.withColumn('Year', lit(2022))
```

In [8]:
```
#Ensure every record can be uniquely identified
#combine fifa id and year as index
from pyspark.sql import functions as sf

players_15 = players_15.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
players_16 = players_16.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
players_17 = players_17.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
```

```
players_18 = players_18.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
players_19 = players_19.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
players_20 = players_20.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
players_21 = players_21.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
players_22 = players_22.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col
```

In [9]:

```
# union the data to one dataset
data = players_15.union(players_16)
data = data.union(players_17)
data = data.union(players_18)
data = data.union(players_19)
data = data.union(players_20)
data = data.union(players_21)
data = data.union(players_22)
```

In [13]:

```
# db_properties={}
# #update your db username
# db_properties['username']="postgres"
# #update your db password
# db_properties['password']="psql"
# #make sure you got the right port number here
# db_properties['url']= "jdbc:postgresql://localhost:5432/postgres"
# #make sure you had the Postgres JAR file in the right location
# db_properties['driver']="org.postgresql.Driver"
# db_properties['table']= "fifa.data"



# data.write.format("jdbc")\
# .mode("overwrite")\
# .option("url", db_properties['url'])\
# .option("dbtable", db_properties['table'])\
# .option("user", db_properties['username'])\
# .option("password", db_properties['password'])\
# .option("Driver", db_properties['driver'])\
# .save()
```

In [14]:

```
data.printSchema()
```

```
root
 |-- sofifa_id: integer (nullable = true)
 |-- player_url: string (nullable = true)
 |-- short_name: string (nullable = true)
 |-- long_name: string (nullable = true)
 |-- player_positions: string (nullable = true)
 |-- overall: integer (nullable = true)
 |-- potential: integer (nullable = true)
 |-- value_eur: double (nullable = true)
 |-- wage_eur: double (nullable = true)
 |-- age: integer (nullable = true)
 |-- dob: string (nullable = true)
 |-- height_cm: integer (nullable = true)
 |-- weight_kg: integer (nullable = true)
 |-- club_team_id: double (nullable = true)
 |-- club_name: string (nullable = true)
 |-- league_name: string (nullable = true)
 |-- league_level: integer (nullable = true)
 |-- club_position: string (nullable = true)
 |-- club_jersey_number: integer (nullable = true)
 |-- club_loaned_from: string (nullable = true)
 |-- club_joined: string (nullable = true)
 |-- club_contract_valid_until: integer (nullable = true)
 |-- nationality_id: integer (nullable = true)
 |-- nationality_name: string (nullable = true)
 |-- nation_team_id: double (nullable = true)
 |-- nation_position: string (nullable = true)
 |-- nation_jersey_number: integer (nullable = true)
```

```
|-- preferred_foot: string (nullable = true)
|-- weak_foot: integer (nullable = true)
|-- skill_moves: integer (nullable = true)
|-- international_reputation: integer (nullable = true)
|-- work_rate: string (nullable = true)
|-- body_type: string (nullable = true)
|-- real_face: string (nullable = true)
|-- release_clause_eur: string (nullable = true)
|-- player_tags: string (nullable = true)
|-- player_traits: string (nullable = true)
|-- pace: integer (nullable = true)
|-- shooting: integer (nullable = true)
|-- passing: integer (nullable = true)
|-- dribbling: integer (nullable = true)
|-- defending: integer (nullable = true)
|-- physic: integer (nullable = true)
|-- attacking_crossing: integer (nullable = true)
|-- attacking_finishing: integer (nullable = true)
|-- attacking_heading_accuracy: integer (nullable = true)
|-- attacking_short_passing: integer (nullable = true)
|-- attacking_volleys: integer (nullable = true)
|-- skill_dribbling: integer (nullable = true)
|-- skill_curve: integer (nullable = true)
|-- skill_fk_accuracy: integer (nullable = true)
|-- skill_long_passing: integer (nullable = true)
|-- skill_ball_control: integer (nullable = true)
|-- movement_acceleration: integer (nullable = true)
|-- movement_sprint_speed: integer (nullable = true)
|-- movement_agility: integer (nullable = true)
|-- movement_reactions: integer (nullable = true)
|-- movement_balance: integer (nullable = true)
|-- power_shot_power: integer (nullable = true)
|-- power_jumping: integer (nullable = true)
|-- power_stamina: integer (nullable = true)
|-- power_strength: integer (nullable = true)
|-- power_long_shots: integer (nullable = true)
|-- mentality_aggression: integer (nullable = true)
|-- mentality_interceptions: integer (nullable = true)
|-- mentality_positioning: integer (nullable = true)
|-- mentality_vision: integer (nullable = true)
|-- mentality_penalties: integer (nullable = true)
|-- mentality_composure: string (nullable = true)
|-- defending_marking_awareness: integer (nullable = true)
|-- defending_standing_tackle: integer (nullable = true)
|-- defending_sliding_tackle: integer (nullable = true)
|-- goalkeeping_diving: integer (nullable = true)
|-- goalkeeping_handling: integer (nullable = true)
|-- goalkeeping_kicking: integer (nullable = true)
|-- goalkeeping_positioning: integer (nullable = true)
|-- goalkeeping_reflexes: integer (nullable = true)
|-- goalkeeping_speed: integer (nullable = true)
|-- ls: string (nullable = true)
|-- st: string (nullable = true)
|-- rs: string (nullable = true)
|-- lw: string (nullable = true)
|-- lf: string (nullable = true)
|-- cf: string (nullable = true)
|-- rf: string (nullable = true)
|-- rw: string (nullable = true)
|-- lam: string (nullable = true)
|-- cam: string (nullable = true)
|-- ram: string (nullable = true)
|-- lm: string (nullable = true)
|-- lcm: string (nullable = true)
|-- cm: string (nullable = true)
|-- rcm: string (nullable = true)
|-- rm: string (nullable = true)
|-- lwb: string (nullable = true)
|-- ldm: string (nullable = true)
|-- cdm: string (nullable = true)
|-- rdm: string (nullable = true)
|-- rwb: string (nullable = true)
|-- lb: string (nullable = true)
|-- lcb: string (nullable = true)
```

```
|-- cb: string (nullable = true)
|-- rcb: string (nullable = true)
|-- rb: string (nullable = true)
|-- gk: string (nullable = true)
|-- player_face_url: string (nullable = true)
|-- club_logo_url: string (nullable = true)
|-- club_flag_url: string (nullable = true)
|-- nation_logo_url: string (nullable = true)
|-- nation_flag_url: string (nullable = true)
|-- Year: integer (nullable = false)
|-- data_ID: string (nullable = true)
```

In [15]:

```
# # read the data back from the postgres
# data_read = sqlContext.read.format("jdbc")\
#     .option("url", db_properties['url'])\
#     .option("dbtable", db_properties['table'])\
#     .option("user", db_properties['username'])\
#     .option("password", db_properties['password'])\
#     .option("Driver", db_properties['driver'])\
#     .load()

# data_read.show(1, vertical=True)
```

In [17]:

```
data_read = data
```

Task II

In [19]:

```
#In order to find the club with highest numbers of players
#use where to find the data in 2022
#group the data by club name and sort them by the count of data
highest_number_club = data_read.where(data_read['Year']=='2022').groupBy("club_name").count().s
print("club with highest numbers:",highest_number_club['club_name'].values)
```

```
club with highest numbers: ['TSG Hoffenheim' 'Paris Saint-Germain' 'Newcastle United'
 'Crystal Palace' 'Arsenal' 'Burnley' 'RCD Espanyol de Barcelona'
 'RCD Mallorca' 'RC Celta de Vigo' 'Manchester United' 'Brentford'
 'Granada CF' 'Genoa' 'Real Madrid CF' 'ESTAC Troyes' 'FC Barcelona'
 'VfB Stuttgart' 'Villarreal CF' 'Real Betis Balompié' 'CA Osasuna'
 'Southampton' 'Leicester City' 'Venezia FC' 'Valencia CF'
 'Borussia Mönchengladbach' 'Tottenham Hotspur' 'Sevilla FC'
 'Wolverhampton Wanderers' 'Brighton & Hove Albion' 'Liverpool'
 'Norwich City' 'Olympique de Marseille' 'Chelsea' 'Everton'
 'West Ham United']
```

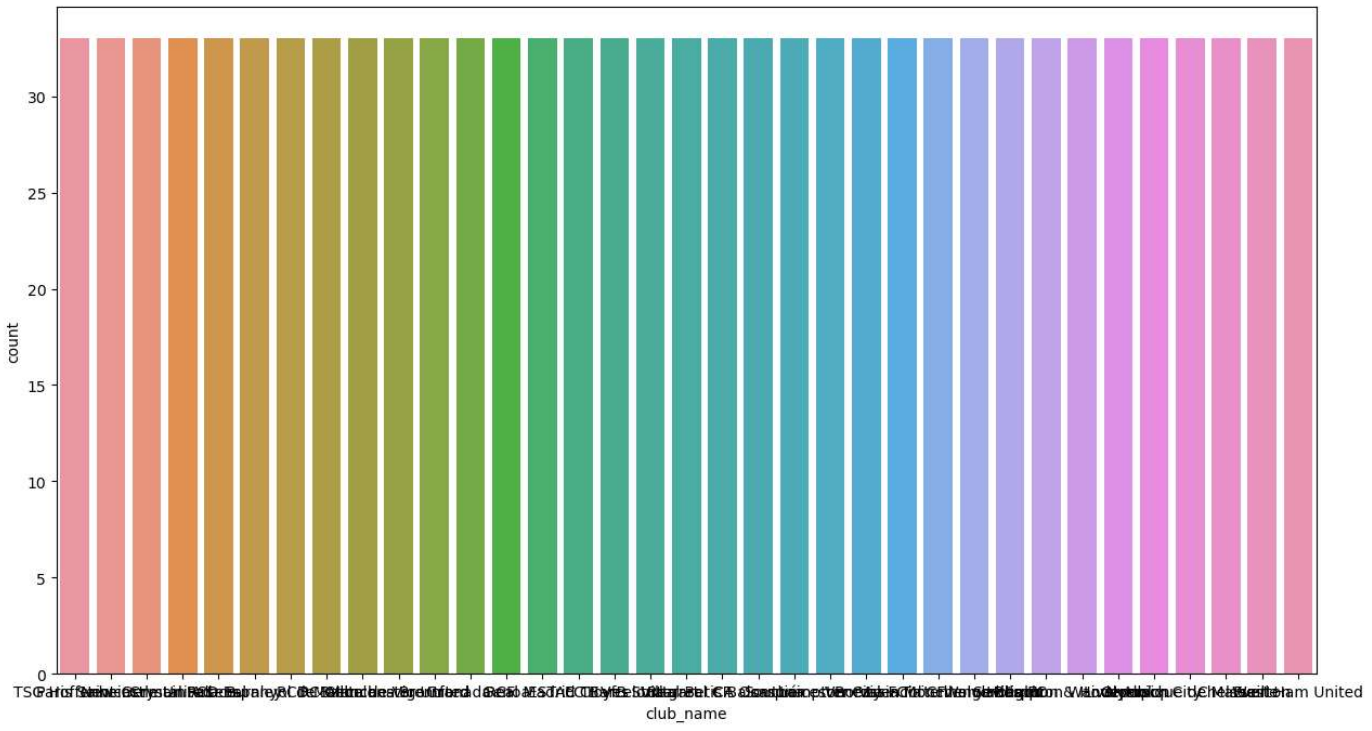
In [20]:

```
# this kernel shows the detailed number of players in each club and we can have the highest num
# we drew the barplot of the count to give a direct interpretation
import seaborn as sns
from IPython import display
import matplotlib.pyplot as plt

top_50_highest_number_club = data_read.where(data_read['Year']=='2022').groupBy("club_name").co
print(top_50_highest_number_club)
plt.figure( figsize = ( 15, 8 ) )
sns.barplot( x="club_name", y="count", data=highest_number_club)
plt.show()
```

```
                club_name  count
0                    None     61
1      Manchester United      33
2              Brentford      33
3                Arsenal      33
4           RCD Mallorca      33
5          Real Madrid CF     33
6              Southampton    33
```

7	RC Celta de Vigo	33
8	Venezia FC	33
9	Borussia Mönchengladbach	33
10	Brighton & Hove Albion	33
11	Genoa	33
12	Paris Saint-Germain	33
13	Liverpool	33
14	Burnley	33
15	Norwich City	33
16	Granada CF	33
17	Tottenham Hotspur	33
18	Newcastle United	33
19	Olympique de Marseille	33
20	Sevilla FC	33
21	Wolverhampton Wanderers	33
22	VfB Stuttgart	33
23	West Ham United	33
24	Leicester City	33
25	Real Betis Balompié	33
26	FC Barcelona	33
27	Chelsea	33
28	Valencia CF	33
29	CA Osasuna	33
30	Everton	33
31	Villarreal CF	33
32	ESTAC Troyes	33
33	Crystal Palace	33
34	Levante Unión Deportiva	33
35	TSG Hoffenheim	33
36	RCD Espanyol de Barcelona	33
37	VfL Wolfsburg	32
38	Cádiz CF	32
39	SpVgg Greuther Fürth	32
40	1. FSV Mainz 05	32
41	RB Leipzig	32
42	Lazio	32
43	Manchester City	32
44	Atlético de Madrid	32
45	Hertha BSC	32
46	Roma	32
47	Borussia Dortmund	31
48	Leeds United	31
49	Aston Villa	31



In [21]:

```
# the number of players older than 27 years oldfor each club
# we used where to find the players who are older than 27 and group the data by club_name, ther
# to find the club who has the highest number
# we sum the age for each club
```

```
import numpy as np
highest_number_over27_count = data_read.where(data_read['age']>27).groupBy("club_name").count()
highest_number_over27_sum = data_read.where(data_read['age']>27).groupBy("club_name").sum('age'
# highest_number_over27_value = highest_number_over27['age'].values
# .sort("count",ascending=False).limit(50).toPandas()
print(highest_number_over27_count)
print(highest_number_over27_sum)
average_over27 = highest_number_over27_sum['sum(age)'].values/highest_number_over27_count['cour
print(type(average_over27),average_over27)
```

```
[Stage 51:================================================>       (25 + 4) / 29]
                    club_name  count
0                        None    874
1     İstanbul Başakşehir FK    133
2     Jeonbuk Hyundai Motors    118
3        FC Lokomotiv Moscow    108
4             Crystal Palace    106
...                      ...    ...
1007     Caracas Fútbol Club      3
1008          SC Freiburg II      2
1009            FC Helsingør      2
1010            FC Dordrecht      1
1011    Borussia Dortmund II      1

[1012 rows x 2 columns]
                       club_name  sum(age)
0                        Palermo      1105
1              Santiago Wanderers       749
2              1. FC Union Berlin      2215
3                           Carpi       554
4     Club Independiente Santa Fe      2119
...                           ...       ...
1007                    Gefle IF       614
1008                        Bury      1731
1009           Accrington Stanley      1020
1010                 Bohemian FC      1269
1011                        Como       219

[1012 rows x 2 columns]
<class 'numpy.ndarray'> [1.26430206e+00 5.63157895e+00 1.87711864e+01 ... 5.10000000e+02
 1.26900000e+03 2.19000000e+02]
```

In [22]:

```
# we calculated the average and found the highest average
highest_number_over27_avg = data_read.where(data_read['age']>27).groupBy("club_name").avg('age'
print(highest_number_over27_avg)
```

```
[Stage 54:=========================================>              (22 + 4) / 29]
               club_name   avg(age)
0            Yokohama FC  34.703704
1         Wexford Youths  34.000000
2     Zamora Fútbol Club  33.857143
3  Centro Atlético Fénix  33.600000
4         CF Fuenlabrada  33.545455
```

In [23]:

```
top1_highest_number_over27_avg = data_read.where(data_read['age']>27).groupBy("club_name").avg(
print("club with highest numbers over 27:",top1_highest_number_over27_avg['club_name'].values)
```

```
[Stage 57:================================================>       (25 + 4) / 29]
club with highest numbers over 27: ['Yokohama FC']
```

In [24]:

```
# most frequent nation_positions for each year
# we also used where to filter the year, and group them by nation_position, count the number of
# them to search the highest one.
most_frequent_nation_2015 = data_read.where(data_read['Year']==2015).groupBy("nation_position")
print("Most frequent nation position 2015:",most_frequent_nation_2015['nation_position'].values
```

```
most_frequent_nation_2016 = data_read.where(data_read['Year']==2016).groupBy("nation_position")
print("Most frequent nation position 2016:",most_frequent_nation_2016['nation_position'].values
most_frequent_nation_2017 = data_read.where(data_read['Year']==2017).groupBy("nation_position")
print("Most frequent nation position 2017:",most_frequent_nation_2017['nation_position'].values
most_frequent_nation_2018 = data_read.where(data_read['Year']==2018).groupBy("nation_position")
print("Most frequent nation position 2018:",most_frequent_nation_2018['nation_position'].values
most_frequent_nation_2019 = data_read.where(data_read['Year']==2019).groupBy("nation_position")
print("Most frequent nation position 2019:",most_frequent_nation_2019['nation_position'].values
most_frequent_nation_2020 = data_read.where(data_read['Year']==2020).groupBy("nation_position")
print("Most frequent nation position 2020:",most_frequent_nation_2020['nation_position'].values
most_frequent_nation_2021 = data_read.where(data_read['Year']==2021).groupBy("nation_position")
print("Most frequent nation position 2021:",most_frequent_nation_2021['nation_position'].values
most_frequent_nation_2022 = data_read.where(data_read['Year']==2022).groupBy("nation_position")
print("Most frequent nation position 2022:",most_frequent_nation_2022['nation_position'].values
```

```
Most frequent nation position 2015: ['SUB']
Most frequent nation position 2016: ['SUB']
Most frequent nation position 2017: ['SUB']
Most frequent nation position 2018: ['SUB']
Most frequent nation position 2019: ['SUB']
Most frequent nation position 2020: ['SUB']
Most frequent nation position 2021: ['SUB']
Most frequent nation position 2022: ['SUB']
```

In [25]:

```
# most_frequent_nation_2015 = data_read.where(data_read['Year']==2015).groupBy("nation_positior
# print(most_frequent_nation_2015)
```

Task III

Data Engineering and cleaning

In [26]:

```
# Since we only use the skillsets of the players, we dropped all the unneccessary columns
drop_list = ['player_position',' nationality_name','work_rate','body_type','real_face','data_ID
                            'club_loaned_from','club_joined','club_contract_valid_until',
                            'nationality_id','nation_team_id','nation_position','nation_jersey_
                            'release_clause_eur','player_tags','player_traits','player_face_url
                            'club_logo_url','club_flag_url','nation_logo_url','nation_flag_url'
                            'player_url','mentality_composure','goalkeeping_speed','club_positi
```

In [27]:

```
print(len(drop_list))
```

```
33
```

In [28]:

```
df_read = data_read.drop('player_positions','nationality_name','work_rate','body_type','real_fa
                            'club_loaned_from','club_joined','club_contract_valid_until',
                            'nationality_id','nation_team_id','nation_position','nation_jersey_
                            'release_clause_eur','player_tags','player_traits','player_face_url
                            'club_logo_url','club_flag_url','nation_logo_url','nation_flag_url'
                            'player_url','mentality_composure','goalkeeping_speed','club_positi
```

In [29]:

```
print(len(data_read.columns))
print(len(df_read.columns))
```

```
112
79
```

In [30]:

```
df_read.printSchema()
```

```
root
 |-- sofifa_id: integer (nullable = true)
 |-- overall: integer (nullable = true)
```

```
|-- potential: integer (nullable = true)
|-- value_eur: double (nullable = true)
|-- wage_eur: double (nullable = true)
|-- age: integer (nullable = true)
|-- height_cm: integer (nullable = true)
|-- weight_kg: integer (nullable = true)
|-- preferred_foot: string (nullable = true)
|-- weak_foot: integer (nullable = true)
|-- skill_moves: integer (nullable = true)
|-- international_reputation: integer (nullable = true)
|-- pace: integer (nullable = true)
|-- shooting: integer (nullable = true)
|-- passing: integer (nullable = true)
|-- dribbling: integer (nullable = true)
|-- defending: integer (nullable = true)
|-- physic: integer (nullable = true)
|-- attacking_crossing: integer (nullable = true)
|-- attacking_finishing: integer (nullable = true)
|-- attacking_heading_accuracy: integer (nullable = true)
|-- attacking_short_passing: integer (nullable = true)
|-- attacking_volleys: integer (nullable = true)
|-- skill_dribbling: integer (nullable = true)
|-- skill_curve: integer (nullable = true)
|-- skill_fk_accuracy: integer (nullable = true)
|-- skill_long_passing: integer (nullable = true)
|-- skill_ball_control: integer (nullable = true)
|-- movement_acceleration: integer (nullable = true)
|-- movement_sprint_speed: integer (nullable = true)
|-- movement_agility: integer (nullable = true)
|-- movement_reactions: integer (nullable = true)
|-- movement_balance: integer (nullable = true)
|-- power_shot_power: integer (nullable = true)
|-- power_jumping: integer (nullable = true)
|-- power_stamina: integer (nullable = true)
|-- power_strength: integer (nullable = true)
|-- power_long_shots: integer (nullable = true)
|-- mentality_aggression: integer (nullable = true)
|-- mentality_interceptions: integer (nullable = true)
|-- mentality_positioning: integer (nullable = true)
|-- mentality_vision: integer (nullable = true)
|-- mentality_penalties: integer (nullable = true)
|-- defending_marking_awareness: integer (nullable = true)
|-- defending_standing_tackle: integer (nullable = true)
|-- defending_sliding_tackle: integer (nullable = true)
|-- goalkeeping_diving: integer (nullable = true)
|-- goalkeeping_handling: integer (nullable = true)
|-- goalkeeping_kicking: integer (nullable = true)
|-- goalkeeping_positioning: integer (nullable = true)
|-- goalkeeping_reflexes: integer (nullable = true)
|-- ls: string (nullable = true)
|-- st: string (nullable = true)
|-- rs: string (nullable = true)
|-- lw: string (nullable = true)
|-- lf: string (nullable = true)
|-- cf: string (nullable = true)
|-- rf: string (nullable = true)
|-- rw: string (nullable = true)
|-- lam: string (nullable = true)
|-- cam: string (nullable = true)
|-- ram: string (nullable = true)
|-- lm: string (nullable = true)
|-- lcm: string (nullable = true)
|-- cm: string (nullable = true)
|-- rcm: string (nullable = true)
|-- rm: string (nullable = true)
|-- lwb: string (nullable = true)
|-- ldm: string (nullable = true)
|-- cdm: string (nullable = true)
|-- rdm: string (nullable = true)
|-- rwb: string (nullable = true)
|-- lb: string (nullable = true)
|-- lcb: string (nullable = true)
|-- cb: string (nullable = true)
|-- rcb: string (nullable = true)
```

```
|-- rb: string (nullable = true)
|-- gk: string (nullable = true)
|-- Year: integer (nullable = false)
```

In [31]:

```
# from pyspark.sql.functions import *

# # Notice I dropped isPenalty and isSTPlay
# null_counts_plays_df = df_read.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) \
#                         for c in df_read.columns])

# null_counts_plays_df.show(truncate=False, vertical=True)
```

impute 'pace','shooting','passing','dribbling','defending','physic'

In [32]:

```
# we checked the missing values of the whole dataset and we found out that the columns with
# larger amoungt of missing value: 'pace','shooting','passing','dribbling','defending','physic
# Because there were many missing value in the data, we want to impute the missing value with t
```

In [33]:

```
from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['pace']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na = df_read.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
            inputCols=columns_to_be_imputed,
            outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
            .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed = imputer.fit(df_with_filled_na).transform(df_with_filled_na)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced = df_imputed.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed = df_imputed_enhanced.withColumnRenamed("pace_imputed","pace")
```

In [34]:

```
from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['shooting']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na2 = df_fully_imputed.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
            inputCols=columns_to_be_imputed,
            outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
            .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed2 = imputer.fit(df_with_filled_na2).transform(df_with_filled_na2)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced2 = df_imputed2.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed2 = df_imputed_enhanced2.withColumnRenamed("shooting_imputed","shooting")
```

In [35]:

```
from pyspark.ml.feature import Imputer
```

```
columns_to_be_imputed = ['passing']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na3 = df_fully_imputed2.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
            inputCols=columns_to_be_imputed,
            outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
            .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed3 = imputer.fit(df_with_filled_na3).transform(df_with_filled_na3)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced3 = df_imputed3.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed3 = df_imputed_enhanced3.withColumnRenamed("passing_imputed","passing")
```

In [36]:

```
from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['dribbling']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na4 = df_fully_imputed3.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
            inputCols=columns_to_be_imputed,
            outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
            .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed4 = imputer.fit(df_with_filled_na4).transform(df_with_filled_na4)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced4 = df_imputed4.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed4 = df_imputed_enhanced4.withColumnRenamed("dribbling_imputed","dribbling")
```

In [37]:

```
from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['defending']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na5 = df_fully_imputed4.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
            inputCols=columns_to_be_imputed,
            outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
            .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed5 = imputer.fit(df_with_filled_na5).transform(df_with_filled_na5)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced5 = df_imputed5.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed5 = df_imputed_enhanced5.withColumnRenamed("defending_imputed","defending")
```

In [38]:

```
from pyspark.ml.feature import Imputer
```

```
columns_to_be_imputed = ['physic']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na6 = df_fully_imputed5.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
            inputCols=columns_to_be_imputed,
            outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
            .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed6 = imputer.fit(df_with_filled_na6).transform(df_with_filled_na6)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced6 = df_imputed6.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed6 = df_imputed_enhanced6.withColumnRenamed("physic_imputed","physic")
```

In [39]:

```
from pyspark.sql.functions import *

# # Notice I dropped isPenalty and isSTPlay
# null_counts_plays_df = df_fully_imputed6.select([count(when(isnan(c) | col(c).isNull(), c)).c
#                         for c in df_read.columns])

# null_counts_plays_df.show(truncate=False, vertical=True)
```

drop null

In [40]:

```
# we chaecked the missing value again and found out the rows with missing value
# Then we drop the rows
# Reason: The number of row was small compared to the entire dataset, so we just drop them
```

In [41]:

```
df_read = df_fully_imputed6.dropna()
```

In [42]:

```
null_counts_plays_df = df_read.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) \
                        for c in df_read.columns])

null_counts_plays_df.show(truncate=False, vertical=True)
```

```
-RECORD 0--------------------------
 sofifa_id                   | 0
 overall                     | 0
 potential                   | 0
 value_eur                   | 0
 wage_eur                    | 0
 age                         | 0
 height_cm                   | 0
 weight_kg                   | 0
 preferred_foot              | 0
 weak_foot                   | 0
 skill_moves                 | 0
 international_reputation    | 0
 attacking_crossing          | 0
 attacking_finishing         | 0
 attacking_heading_accuracy  | 0
 attacking_short_passing     | 0
 attacking_volleys           | 0
 skill_dribbling             | 0
 skill_curve                 | 0
 skill_fk_accuracy           | 0
 skill_long_passing          | 0
```

```
skill_ball_control          | 0
movement_acceleration       | 0
movement_sprint_speed       | 0
movement_agility            | 0
movement_reactions          | 0
movement_balance            | 0
power_shot_power            | 0
power_jumping               | 0
power_stamina               | 0
power_strength              | 0
power_long_shots            | 0
mentality_aggression        | 0
mentality_interceptions     | 0
mentality_positioning       | 0
mentality_vision            | 0
mentality_penalties         | 0
defending_marking_awareness | 0
defending_standing_tackle   | 0
defending_sliding_tackle    | 0
goalkeeping_diving          | 0
goalkeeping_handling        | 0
goalkeeping_kicking         | 0
goalkeeping_positioning     | 0
goalkeeping_reflexes        | 0
ls                          | 0
st                          | 0
rs                          | 0
lw                          | 0
lf                          | 0
cf                          | 0
rf                          | 0
rw                          | 0
lam                         | 0
cam                         | 0
ram                         | 0
lm                          | 0
lcm                         | 0
cm                          | 0
rcm                         | 0
rm                          | 0
lwb                         | 0
ldm                         | 0
cdm                         | 0
rdm                         | 0
rwb                         | 0
lb                          | 0
lcb                         | 0
cb                          | 0
rcb                         | 0
rb                          | 0
gk                          | 0
Year                        | 0
pace                        | 0
shooting                    | 0
passing                     | 0
dribbling                   | 0
defending                   | 0
physic                      | 0
```

Data preprocessing

label to binary

In [43]:
```
# Because there are only two values in the 'preferred_foot' column, we convert the value to 0 c
label_to_binary = udf(lambda name: 0.0 if name == 'Left' else 1.0)
df_read_bi = df_read.withColumn('preferred_foot', label_to_binary(col('preferred_foot')))
```

In [44]:
```
# df_read_bi.show(vertical = True)
```

string to int

In [45]:
```
# We found the format "XX+X" in some columns and we calculated them and converted them to int
def addall(string):
    if len(string)== 4:
        string = int(string[0:2])+int(string[-1])
    elif len(string) ==2:
        string = int(string)
    elif len(string)== 6:
        string = int(string[-2:])
    return string
```

In [46]:
```
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType
```

In [47]:
```
addallUDF = udf(lambda i:addall(i), IntegerType())
```

In [48]:
```
str_to_int_list = ['ls','st','rs','lw','lf','cf','rf','rw','lam','cam','ram','lm','lcm','cm','r
                   'ldm','cdm','rdm','rwb','lb','lcb','cb','rcb','rb','gk']
```

In [49]:
```
df_read_int = df_read_bi.withColumn('ls', addallUDF(col('ls')))
df_read_int = df_read_int.withColumn('st', addallUDF(col('st')))
df_read_int = df_read_int.withColumn('rs', addallUDF(col('rs')))
df_read_int = df_read_int.withColumn('lw', addallUDF(col('lw')))
df_read_int = df_read_int.withColumn('lf', addallUDF(col('lf')))
df_read_int = df_read_int.withColumn('cf', addallUDF(col('cf')))
df_read_int = df_read_int.withColumn('rf', addallUDF(col('rf')))
df_read_int = df_read_int.withColumn('rw', addallUDF(col('rw')))
df_read_int = df_read_int.withColumn('lam', addallUDF(col('lam')))
df_read_int = df_read_int.withColumn('cam', addallUDF(col('cam')))
df_read_int = df_read_int.withColumn('ram', addallUDF(col('ram')))
df_read_int = df_read_int.withColumn('lm', addallUDF(col('lm')))
df_read_int = df_read_int.withColumn('lcm', addallUDF(col('lcm')))
df_read_int = df_read_int.withColumn('cm', addallUDF(col('cm')))
df_read_int = df_read_int.withColumn('rcm', addallUDF(col('rcm')))
df_read_int = df_read_int.withColumn('rm', addallUDF(col('rm')))
df_read_int = df_read_int.withColumn('lwb', addallUDF(col('lwb')))
df_read_int = df_read_int.withColumn('ldm', addallUDF(col('ldm')))
df_read_int = df_read_int.withColumn('cdm', addallUDF(col('cdm')))
df_read_int = df_read_int.withColumn('rdm', addallUDF(col('rdm')))
df_read_int = df_read_int.withColumn('rwb', addallUDF(col('rwb')))
df_read_int = df_read_int.withColumn('lb', addallUDF(col('lb')))
df_read_int = df_read_int.withColumn('lcb', addallUDF(col('lcb')))
df_read_int = df_read_int.withColumn('cb', addallUDF(col('cb')))
df_read_int = df_read_int.withColumn('rcb', addallUDF(col('rcb')))
df_read_int = df_read_int.withColumn('rb', addallUDF(col('rb')))
df_read_int = df_read_int.withColumn('gk', addallUDF(col('gk')))
```

cast data type

In [50]:
```
print(df_read_int.columns)
```

```
['sofifa_id', 'overall', 'potential', 'value_eur', 'wage_eur', 'age', 'height_cm', 'weight_kg',
'preferred_foot', 'weak_foot', 'skill_moves', 'international_reputation', 'attacking_crossing',
'attacking_finishing', 'attacking_heading_accuracy', 'attacking_short_passing', 'attacking_voll
eys', 'skill_dribbling', 'skill_curve', 'skill_fk_accuracy', 'skill_long_passing', 'skill_ball_
control', 'movement_acceleration', 'movement_sprint_speed', 'movement_agility', 'movement_react
ions', 'movement_balance', 'power_shot_power', 'power_jumping', 'power_stamina', 'power_strengt
h', 'power_long_shots', 'mentality_aggression', 'mentality_interceptions', 'mentality_positioni
ng', 'mentality_vision', 'mentality_penalties', 'defending_marking_awareness', 'defending_stand
ing_tackle', 'defending_sliding_tackle', 'goalkeeping_diving', 'goalkeeping_handling', 'goalkee
ping_kicking', 'goalkeeping_positioning', 'goalkeeping_reflexes', 'ls', 'st', 'rs', 'lw', 'lf',
```

```
'cf', 'rf', 'rw', 'lam', 'cam', 'ram', 'lm', 'lcm', 'cm', 'rcm', 'rm', 'lwb', 'ldm', 'cdm', 'rd
m', 'rwb', 'lb', 'lcb', 'cb', 'rcb', 'rb', 'gk', 'Year', 'pace', 'shooting', 'passing', 'dribbl
ing', 'defending', 'physic']
```

In [51]:

```
# we changed the type of data to DoubleType
```

In [52]:

```
from pyspark.sql.types import DoubleType
for i in df_read_int.columns:
    df_read_int = df_read_int.withColumn(i, df_read_int[i].cast(DoubleType()))
```

In [53]:

```
df_read_int.printSchema()
```

```
root
 |-- sofifa_id: double (nullable = true)
 |-- overall: double (nullable = true)
 |-- potential: double (nullable = true)
 |-- value_eur: double (nullable = true)
 |-- wage_eur: double (nullable = true)
 |-- age: double (nullable = true)
 |-- height_cm: double (nullable = true)
 |-- weight_kg: double (nullable = true)
 |-- preferred_foot: double (nullable = true)
 |-- weak_foot: double (nullable = true)
 |-- skill_moves: double (nullable = true)
 |-- international_reputation: double (nullable = true)
 |-- attacking_crossing: double (nullable = true)
 |-- attacking_finishing: double (nullable = true)
 |-- attacking_heading_accuracy: double (nullable = true)
 |-- attacking_short_passing: double (nullable = true)
 |-- attacking_volleys: double (nullable = true)
 |-- skill_dribbling: double (nullable = true)
 |-- skill_curve: double (nullable = true)
 |-- skill_fk_accuracy: double (nullable = true)
 |-- skill_long_passing: double (nullable = true)
 |-- skill_ball_control: double (nullable = true)
 |-- movement_acceleration: double (nullable = true)
 |-- movement_sprint_speed: double (nullable = true)
 |-- movement_agility: double (nullable = true)
 |-- movement_reactions: double (nullable = true)
 |-- movement_balance: double (nullable = true)
 |-- power_shot_power: double (nullable = true)
 |-- power_jumping: double (nullable = true)
 |-- power_stamina: double (nullable = true)
 |-- power_strength: double (nullable = true)
 |-- power_long_shots: double (nullable = true)
 |-- mentality_aggression: double (nullable = true)
 |-- mentality_interceptions: double (nullable = true)
 |-- mentality_positioning: double (nullable = true)
 |-- mentality_vision: double (nullable = true)
 |-- mentality_penalties: double (nullable = true)
 |-- defending_marking_awareness: double (nullable = true)
 |-- defending_standing_tackle: double (nullable = true)
 |-- defending_sliding_tackle: double (nullable = true)
 |-- goalkeeping_diving: double (nullable = true)
 |-- goalkeeping_handling: double (nullable = true)
 |-- goalkeeping_kicking: double (nullable = true)
 |-- goalkeeping_positioning: double (nullable = true)
 |-- goalkeeping_reflexes: double (nullable = true)
 |-- ls: double (nullable = true)
 |-- st: double (nullable = true)
 |-- rs: double (nullable = true)
 |-- lw: double (nullable = true)
 |-- lf: double (nullable = true)
 |-- cf: double (nullable = true)
 |-- rf: double (nullable = true)
 |-- rw: double (nullable = true)
 |-- lam: double (nullable = true)
 |-- cam: double (nullable = true)
 |-- ram: double (nullable = true)
 |-- lm: double (nullable = true)
```

```
|-- lcm: double (nullable = true)
|-- cm: double (nullable = true)
|-- rcm: double (nullable = true)
|-- rm: double (nullable = true)
|-- lwb: double (nullable = true)
|-- ldm: double (nullable = true)
|-- cdm: double (nullable = true)
|-- rdm: double (nullable = true)
|-- rwb: double (nullable = true)
|-- lb: double (nullable = true)
|-- lcb: double (nullable = true)
|-- cb: double (nullable = true)
|-- rcb: double (nullable = true)
|-- rb: double (nullable = true)
|-- gk: double (nullable = true)
|-- Year: double (nullable = false)
|-- pace: double (nullable = true)
|-- shooting: double (nullable = true)
|-- passing: double (nullable = true)
|-- dribbling: double (nullable = true)
|-- defending: double (nullable = true)
|-- physic: double (nullable = true)
```

In [54]:

```
df_read_features = df_read_int.withColumn('overall_new',col('overall')).drop('overall')
```

In [55]:

```
df_read_int.show(5,vertical = True)
```

```
[Stage 120:>                                                        (0 + 1) / 1]
-RECORD 0------------------------------
 sofifa_id                    | 158023.0
 overall                      | 93.0
 potential                    | 95.0
 value_eur                    | 1.005E8
 wage_eur                     | 550000.0
 age                          | 27.0
 height_cm                    | 169.0
 weight_kg                    | 67.0
 preferred_foot               | 0.0
 weak_foot                    | 3.0
 skill_moves                  | 4.0
 international_reputation     | 5.0
 attacking_crossing           | 84.0
 attacking_finishing          | 94.0
 attacking_heading_accuracy   | 71.0
 attacking_short_passing      | 89.0
 attacking_volleys            | 85.0
 skill_dribbling              | 96.0
 skill_curve                  | 89.0
 skill_fk_accuracy            | 90.0
 skill_long_passing           | 76.0
 skill_ball_control           | 96.0
 movement_acceleration        | 96.0
 movement_sprint_speed        | 90.0
 movement_agility             | 94.0
 movement_reactions           | 94.0
 movement_balance             | 95.0
 power_shot_power             | 80.0
 power_jumping                | 73.0
 power_stamina                | 77.0
 power_strength               | 60.0
 power_long_shots             | 88.0
 mentality_aggression         | 48.0
 mentality_interceptions      | 22.0
 mentality_positioning        | 92.0
 mentality_vision             | 90.0
 mentality_penalties          | 76.0
 defending_marking_awareness  | 25.0
 defending_standing_tackle    | 21.0
 defending_sliding_tackle     | 20.0
 goalkeeping_diving           | 6.0
```

```
goalkeeping_handling        | 11.0
goalkeeping_kicking         | 15.0
goalkeeping_positioning     | 14.0
goalkeeping_reflexes        | 8.0
ls                          | 92.0
st                          | 92.0
rs                          | 92.0
lw                          | 95.0
lf                          | 93.0
cf                          | 93.0
rf                          | 93.0
rw                          | 95.0
lam                         | 95.0
cam                         | 95.0
ram                         | 95.0
lm                          | 93.0
lcm                         | 82.0
cm                          | 82.0
rcm                         | 82.0
rm                          | 93.0
lwb                         | 65.0
ldm                         | 65.0
cdm                         | 65.0
rdm                         | 65.0
rwb                         | 65.0
lb                          | 57.0
lcb                         | 48.0
cb                          | 48.0
rcb                         | 48.0
rb                          | 57.0
gk                          | 18.0
Year                        | 2015.0
pace                        | 93.0
shooting                    | 89.0
passing                     | 86.0
dribbling                   | 96.0
defending                   | 27.0
physic                      | 63.0
-RECORD 1-----------------------------
sofifa_id                   | 20801.0
overall                     | 92.0
potential                   | 92.0
value_eur                   | 7.9E7
wage_eur                    | 375000.0
age                         | 29.0
height_cm                   | 185.0
weight_kg                   | 80.0
preferred_foot              | 1.0
weak_foot                   | 4.0
skill_moves                 | 5.0
international_reputation    | 5.0
attacking_crossing          | 83.0
attacking_finishing         | 95.0
attacking_heading_accuracy  | 86.0
attacking_short_passing     | 82.0
attacking_volleys           | 87.0
skill_dribbling             | 93.0
skill_curve                 | 88.0
skill_fk_accuracy           | 79.0
skill_long_passing          | 72.0
skill_ball_control          | 92.0
movement_acceleration       | 91.0
movement_sprint_speed       | 94.0
movement_agility            | 93.0
movement_reactions          | 90.0
movement_balance            | 63.0
power_shot_power            | 94.0
power_jumping               | 94.0
power_stamina               | 89.0
power_strength              | 79.0
power_long_shots            | 93.0
mentality_aggression        | 63.0
mentality_interceptions     | 24.0
mentality_positioning       | 91.0
```

```
 mentality_vision            | 81.0
 mentality_penalties         | 85.0
 defending_marking_awareness | 22.0
 defending_standing_tackle   | 31.0
 defending_sliding_tackle    | 23.0
 goalkeeping_diving          | 7.0
 goalkeeping_handling        | 11.0
 goalkeeping_kicking         | 15.0
 goalkeeping_positioning     | 14.0
 goalkeeping_reflexes        | 11.0
 ls                          | 92.0
 st                          | 92.0
 rs                          | 92.0
 lw                          | 92.0
 lf                          | 92.0
 cf                          | 92.0
 rf                          | 92.0
 rw                          | 92.0
 lam                         | 92.0
 cam                         | 92.0
 ram                         | 92.0
 lm                          | 90.0
 lcm                         | 80.0
 cm                          | 80.0
 rcm                         | 80.0
 rm                          | 90.0
 lwb                         | 66.0
 ldm                         | 66.0
 cdm                         | 66.0
 rdm                         | 66.0
 rwb                         | 66.0
 lb                          | 60.0
 lcb                         | 55.0
 cb                          | 55.0
 rcb                         | 55.0
 rb                          | 60.0
 gk                          | 19.0
 Year                        | 2015.0
 pace                        | 93.0
 shooting                    | 93.0
 passing                     | 81.0
 dribbling                   | 91.0
 defending                   | 32.0
 physic                      | 79.0
-RECORD 2-----------------------------
 sofifa_id                   | 9014.0
 overall                     | 90.0
 potential                   | 90.0
 value_eur                   | 5.45E7
 wage_eur                    | 275000.0
 age                         | 30.0
 height_cm                   | 180.0
 weight_kg                   | 80.0
 preferred_foot              | 0.0
 weak_foot                   | 2.0
 skill_moves                 | 4.0
 international_reputation    | 5.0
 attacking_crossing          | 80.0
 attacking_finishing         | 85.0
 attacking_heading_accuracy  | 50.0
 attacking_short_passing     | 86.0
 attacking_volleys           | 86.0
 skill_dribbling             | 93.0
 skill_curve                 | 85.0
 skill_fk_accuracy           | 83.0
 skill_long_passing          | 76.0
 skill_ball_control          | 90.0
 movement_acceleration       | 93.0
 movement_sprint_speed       | 93.0
 movement_agility            | 93.0
 movement_reactions          | 89.0
 movement_balance            | 91.0
 power_shot_power            | 86.0
 power_jumping               | 61.0
```

```
 power_stamina                | 78.0
 power_strength               | 65.0
 power_long_shots             | 90.0
 mentality_aggression         | 47.0
 mentality_interceptions      | 39.0
 mentality_positioning        | 89.0
 mentality_vision             | 84.0
 mentality_penalties          | 80.0
 defending_marking_awareness  | 29.0
 defending_standing_tackle    | 26.0
 defending_sliding_tackle     | 26.0
 goalkeeping_diving           | 10.0
 goalkeeping_handling         | 8.0
 goalkeeping_kicking          | 11.0
 goalkeeping_positioning      | 5.0
 goalkeeping_reflexes         | 15.0
 ls                           | 87.0
 st                           | 87.0
 rs                           | 87.0
 lw                           | 90.0
 lf                           | 90.0
 cf                           | 90.0
 rf                           | 90.0
 rw                           | 90.0
 lam                          | 90.0
 cam                          | 90.0
 ram                          | 90.0
 lm                           | 90.0
 lcm                          | 81.0
 cm                           | 81.0
 rcm                          | 81.0
 rm                           | 90.0
 lwb                          | 67.0
 ldm                          | 67.0
 cdm                          | 67.0
 rdm                          | 67.0
 rwb                          | 67.0
 lb                           | 58.0
 lcb                          | 49.0
 cb                           | 49.0
 rcb                          | 49.0
 rb                           | 58.0
 gk                           | 17.0
 Year                         | 2015.0
 pace                         | 93.0
 shooting                     | 86.0
 passing                      | 83.0
 dribbling                    | 92.0
 defending                    | 32.0
 physic                       | 64.0
-RECORD 3-----------------------------
 sofifa_id                    | 41236.0
 overall                      | 90.0
 potential                    | 90.0
 value_eur                    | 5.25E7
 wage_eur                     | 275000.0
 age                          | 32.0
 height_cm                    | 195.0
 weight_kg                    | 95.0
 preferred_foot               | 1.0
 weak_foot                    | 4.0
 skill_moves                  | 4.0
 international_reputation     | 5.0
 attacking_crossing           | 76.0
 attacking_finishing          | 91.0
 attacking_heading_accuracy   | 76.0
 attacking_short_passing      | 84.0
 attacking_volleys            | 92.0
 skill_dribbling              | 88.0
 skill_curve                  | 80.0
 skill_fk_accuracy            | 80.0
 skill_long_passing           | 76.0
 skill_ball_control           | 90.0
 movement_acceleration        | 74.0
```

```
 movement_sprint_speed       | 77.0
 movement_agility            | 86.0
 movement_reactions          | 85.0
 movement_balance            | 41.0
 power_shot_power            | 93.0
 power_jumping               | 72.0
 power_stamina               | 78.0
 power_strength              | 93.0
 power_long_shots            | 88.0
 mentality_aggression        | 84.0
 mentality_interceptions     | 20.0
 mentality_positioning       | 86.0
 mentality_vision            | 83.0
 mentality_penalties         | 91.0
 defending_marking_awareness | 25.0
 defending_standing_tackle   | 41.0
 defending_sliding_tackle    | 27.0
 goalkeeping_diving          | 13.0
 goalkeeping_handling        | 15.0
 goalkeeping_kicking         | 10.0
 goalkeeping_positioning     | 9.0
 goalkeeping_reflexes        | 12.0
 ls                          | 90.0
 st                          | 90.0
 rs                          | 90.0
 lw                          | 87.0
 lf                          | 89.0
 cf                          | 89.0
 rf                          | 89.0
 rw                          | 87.0
 lam                         | 89.0
 cam                         | 89.0
 ram                         | 89.0
 lm                          | 86.0
 lcm                         | 79.0
 cm                          | 79.0
 rcm                         | 79.0
 rm                          | 86.0
 lwb                         | 64.0
 ldm                         | 68.0
 cdm                         | 68.0
 rdm                         | 68.0
 rwb                         | 64.0
 lb                          | 59.0
 lcb                         | 58.0
 cb                          | 58.0
 rcb                         | 58.0
 rb                          | 59.0
 gk                          | 20.0
 Year                        | 2015.0
 pace                        | 76.0
 shooting                    | 91.0
 passing                     | 81.0
 dribbling                   | 86.0
 defending                   | 34.0
 physic                      | 86.0
-RECORD 4-----------------------------
 sofifa_id                   | 167495.0
 overall                     | 90.0
 potential                   | 90.0
 value_eur                   | 6.35E7
 wage_eur                    | 300000.0
 age                         | 28.0
 height_cm                   | 193.0
 weight_kg                   | 92.0
 preferred_foot              | 1.0
 weak_foot                   | 4.0
 skill_moves                 | 1.0
 international_reputation    | 5.0
 attacking_crossing          | 25.0
 attacking_finishing         | 25.0
 attacking_heading_accuracy  | 25.0
 attacking_short_passing     | 42.0
 attacking_volleys           | 25.0
```

```
 skill_dribbling              | 25.0
 skill_curve                  | 25.0
 skill_fk_accuracy            | 25.0
 skill_long_passing           | 41.0
 skill_ball_control           | 31.0
 movement_acceleration        | 58.0
 movement_sprint_speed        | 61.0
 movement_agility             | 43.0
 movement_reactions           | 89.0
 movement_balance             | 35.0
 power_shot_power             | 42.0
 power_jumping                | 78.0
 power_stamina                | 44.0
 power_strength               | 83.0
 power_long_shots             | 25.0
 mentality_aggression         | 29.0
 mentality_interceptions      | 30.0
 mentality_positioning        | 25.0
 mentality_vision             | 20.0
 mentality_penalties          | 37.0
 defending_marking_awareness  | 25.0
 defending_standing_tackle    | 25.0
 defending_sliding_tackle     | 25.0
 goalkeeping_diving           | 87.0
 goalkeeping_handling         | 85.0
 goalkeeping_kicking          | 92.0
 goalkeeping_positioning      | 90.0
 goalkeeping_reflexes         | 86.0
 ls                           | 41.0
 st                           | 41.0
 rs                           | 41.0
 lw                           | 39.0
 lf                           | 40.0
 cf                           | 40.0
 rf                           | 40.0
 rw                           | 39.0
 lam                          | 39.0
 cam                          | 39.0
 ram                          | 39.0
 lm                           | 41.0
 lcm                          | 39.0
 cm                           | 39.0
 rcm                          | 39.0
 rm                           | 41.0
 lwb                          | 39.0
 ldm                          | 43.0
 cdm                          | 43.0
 rdm                          | 43.0
 rwb                          | 39.0
 lb                           | 39.0
 lcb                          | 41.0
 cb                           | 41.0
 rcb                          | 41.0
 rb                           | 39.0
 gk                           | 90.0
 Year                         | 2015.0
 pace                         | 69.0
 shooting                     | 54.0
 passing                      | 58.0
 dribbling                    | 63.0
 defending                    | 55.0
 physic                       | 66.0
only showing top 5 rows
```

In [56]:

```
feature_cols = df_read_int.columns
del feature_cols[2]
```

In [57]:

```
from pyspark.sql import Row
from pyspark.ml.linalg import Vectors
```

In [58]:
```
len(df_read_features.columns)
```

Out[58]: 79

In [59]:
```
null_counts_plays_df_2 = df_read_features.select([count(when(isnan(c) | col(c).isNull(), c)).al
                        for c in df_read_features.columns])

null_counts_plays_df_2.show(truncate=False, vertical=True)
```

```
[Stage 121:=================================================>   (27 + 2) / 29]
-RECORD 0-------------------------
 sofifa_id                    | 0
 potential                    | 0
 value_eur                    | 0
 wage_eur                     | 0
 age                          | 0
 height_cm                    | 0
 weight_kg                    | 0
 preferred_foot               | 0
 weak_foot                    | 0
 skill_moves                  | 0
 international_reputation     | 0
 attacking_crossing           | 0
 attacking_finishing          | 0
 attacking_heading_accuracy   | 0
 attacking_short_passing      | 0
 attacking_volleys            | 0
 skill_dribbling              | 0
 skill_curve                  | 0
 skill_fk_accuracy            | 0
 skill_long_passing           | 0
 skill_ball_control           | 0
 movement_acceleration        | 0
 movement_sprint_speed        | 0
 movement_agility             | 0
 movement_reactions           | 0
 movement_balance             | 0
 power_shot_power             | 0
 power_jumping                | 0
 power_stamina                | 0
 power_strength               | 0
 power_long_shots             | 0
 mentality_aggression         | 0
 mentality_interceptions      | 0
 mentality_positioning        | 0
 mentality_vision             | 0
 mentality_penalties          | 0
 defending_marking_awareness  | 0
 defending_standing_tackle    | 0
 defending_sliding_tackle     | 0
 goalkeeping_diving           | 0
 goalkeeping_handling         | 0
 goalkeeping_kicking          | 0
 goalkeeping_positioning      | 0
 goalkeeping_reflexes         | 0
 ls                           | 0
 st                           | 0
 rs                           | 0
 lw                           | 0
 lf                           | 0
 cf                           | 0
 rf                           | 0
 rw                           | 0
 lam                          | 0
 cam                          | 0
 ram                          | 0
 lm                           | 0
 lcm                          | 0
 cm                           | 0
 rcm                          | 0
```

```
rm                          | 0
lwb                         | 0
ldm                         | 0
cdm                         | 0
rdm                         | 0
rwb                         | 0
lb                          | 0
lcb                         | 1
cb                          | 1
rcb                         | 1
rb                          | 0
gk                          | 451
Year                        | 0
pace                        | 0
shooting                    | 0
passing                     | 0
dribbling                   | 0
defending                   | 0
physic                      | 0
overall_new                 | 0
```

In [60]:

```
df_read_features = df_read_features.dropna()
```

In [61]:

```
# we checked the data again to ensure there is no missing value in the dataset
null_counts_plays_df_2 = df_read_features.select([count(when(isnan(c) | col(c).isNull(), c)).al
                        for c in df_read_features.columns])

null_counts_plays_df_2.show(truncate=False, vertical=True)
```

```
[Stage 124:=================================================>   (27 + 2) / 29]
-RECORD 0--------------------------
 sofifa_id                   | 0
 potential                   | 0
 value_eur                   | 0
 wage_eur                    | 0
 age                         | 0
 height_cm                   | 0
 weight_kg                   | 0
 preferred_foot              | 0
 weak_foot                   | 0
 skill_moves                 | 0
 international_reputation    | 0
 attacking_crossing          | 0
 attacking_finishing         | 0
 attacking_heading_accuracy  | 0
 attacking_short_passing     | 0
 attacking_volleys           | 0
 skill_dribbling             | 0
 skill_curve                 | 0
 skill_fk_accuracy           | 0
 skill_long_passing          | 0
 skill_ball_control          | 0
 movement_acceleration       | 0
 movement_sprint_speed       | 0
 movement_agility            | 0
 movement_reactions          | 0
 movement_balance            | 0
 power_shot_power            | 0
 power_jumping               | 0
 power_stamina               | 0
 power_strength              | 0
 power_long_shots            | 0
 mentality_aggression        | 0
 mentality_interceptions     | 0
 mentality_positioning       | 0
 mentality_vision            | 0
 mentality_penalties         | 0
 defending_marking_awareness | 0
 defending_standing_tackle   | 0
```

```
defending_sliding_tackle    | 0
goalkeeping_diving          | 0
goalkeeping_handling        | 0
goalkeeping_kicking         | 0
goalkeeping_positioning     | 0
goalkeeping_reflexes        | 0
ls                          | 0
st                          | 0
rs                          | 0
lw                          | 0
lf                          | 0
cf                          | 0
rf                          | 0
rw                          | 0
lam                         | 0
cam                         | 0
ram                         | 0
lm                          | 0
lcm                         | 0
cm                          | 0
rcm                         | 0
rm                          | 0
lwb                         | 0
ldm                         | 0
cdm                         | 0
rdm                         | 0
rwb                         | 0
lb                          | 0
lcb                         | 0
cb                          | 0
rcb                         | 0
rb                          | 0
gk                          | 0
Year                        | 0
pace                        | 0
shooting                    | 0
passing                     | 0
dribbling                   | 0
defending                   | 0
physic                      | 0
overall_new                 | 0
```

In [62]:

```
# we assembled the features in the data to a vector so that it can be recognized by pyspark mod
```

In [63]:

```
def transData(data):
    return data.rdd.map(lambda r: [r[-1], Vectors.dense(r[:-1])]).\
           toDF(['output','features'])

data= transData(df_read_features)
data.show()
```

```
[Stage 128:>                                                        (0 + 1) / 1]
+------+--------------------+
|output|            features|
+------+--------------------+
|  93.0|[158023.0,95.0,1....|
|  92.0|[20801.0,92.0,7.9...|
|  90.0|[9014.0,90.0,5.45...|
|  90.0|[41236.0,90.0,5.2...|
|  90.0|[167495.0,90.0,6....|
|  89.0|[41.0,89.0,3.6E7,...|
|  89.0|[176580.0,91.0,4....|
|  88.0|[7826.0,88.0,4.05...|
|  88.0|[121944.0,88.0,3....|
|  88.0|[156616.0,88.0,3....|
|  88.0|[167397.0,88.0,4....|
|  88.0|[183277.0,90.0,4....|
|  87.0|[121939.0,87.0,2....|
|  87.0|[155862.0,87.0,3....|
```

```
|  87.0|[164240.0,87.0,2....|
|  87.0|[168542.0,87.0,3....|
|  87.0|[173731.0,91.0,3....|
|  87.0|[177003.0,87.0,3....|
|  87.0|[188545.0,89.0,4....|
|  86.0|[10535.0,86.0,1.5...|
+------+--------------------+
only showing top 20 rows
```

In [64]:
```
# we normalized the features by using StandardScaler
```

In [65]:
```
from pyspark.ml.feature import StandardScaler
Scalerizer=StandardScaler().setInputCol("features").setOutputCol("norm_features")
data_norm = Scalerizer.fit(data).transform(data)
```

In [66]:
```
data_norm = data_norm.drop('features')
data_norm = data_norm.withColumn('features',col('norm_features')).drop('norm_features')
```

In [67]:
```
# data_norm.select('features').show(vertical = False)
```

In [68]:
```
# (trainingData, testData) = data_norm.randomSplit([0.8, 0.2])

# then we splited the data in to training and test dataset

(trainingData, testData) = data_norm.randomSplit([0.8, 0.2])
```

In [69]:
```
trainingData.describe().show()
```

```
[Stage 132:==================================================>   (27 + 2) / 29]
+-------+------------------+
|summary|            output|
+-------+------------------+
|  count|            111785|
|   mean| 65.69243637339535|
| stddev|7.0706646380189575|
|    min|              40.0|
|    max|              94.0|
+-------+------------------+
```

In [70]:
```
print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))
```

```
Training Dataset Count: 111785
[Stage 138:=====================================================> (28 + 1) / 29]
Test Dataset Count: 27945
```

Pyspark

Random Forest

In [71]:

```
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

rf2 = RandomForestRegressor(featuresCol = 'features', labelCol = 'output',numTrees=200, maxDept

# Train model.  This also runs the indexer.
model2 = rf2.fit(trainingData)

# Make predictions.
predictions2 = model2.transform(testData)

predictions2.show(5)
```

```
22/11/28 02:13:47 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
with size 1019.2 KiB
22/11/28 02:13:55 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
with size 1335.4 KiB
22/11/28 02:14:04 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
with size 1965.2 KiB

+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.03820530001504...| 65.3606553081347|
|  66.0|[0.28265544798691...|63.70789157823709|
|  66.0|[0.53014926553508...|66.19326396716272|
|  66.0|[1.32477892358710...|64.28787842304088|
|  66.0|[1.39127585699721...|63.21658468402371|
+------+--------------------+-----------------+
only showing top 5 rows
```

In [72]:

```
evaluator2 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse2 = evaluator2.evaluate(predictions2)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse2)
```

```
[Stage 156:=============================================>         (24 + 4) / 29]
Root Mean Squared Error (RMSE) on test data = 1.57648
```

tune the parameters

In [73]:

```
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

rf3 = RandomForestRegressor(featuresCol = 'features', labelCol = 'output',numTrees=100, maxDept

# Train model.  This also runs the indexer.
model3 = rf3.fit(trainingData)

# Make predictions.
predictions3 = model3.transform(testData)

predictions3.show(5)

evaluator3 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse3 = evaluator3.evaluate(predictions3)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse3)
```

```
22/11/28 02:15:25 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
with size 1039.5 KiB
22/11/28 02:15:29 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
```

```
with size 1355.2 KiB

+------+-------------------+-----------------+
|output|           features|       prediction|
+------+-------------------+-----------------+
|  66.0|[0.03820530001504...| 65.3365914506145|
|  66.0|[0.28265544798691...|63.42295529085955|
|  66.0|[0.53014926553508...|65.97380092447919|
|  66.0|[1.32477892358710...|64.60472831199418|
|  66.0|[1.39127585699721...|63.26268016348623|
+------+-------------------+-----------------+
only showing top 5 rows

[Stage 173:=============================================>         (24 + 4) / 29]
Root Mean Squared Error (RMSE) on test data = 1.56785
```

In [74]:

```
trainingData.show(5)
```

```
+------+-------------------+
|output|           features|
+------+-------------------+
|  66.0|[0.06238073264976...|
|  66.0|[0.09244059313201...|
|  66.0|[0.23534812657222...|
|  66.0|[0.45799980291182...|
|  66.0|[0.52388800240667...|
+------+-------------------+
only showing top 5 rows
```

In [75]:

```
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

rf_1 = RandomForestRegressor(featuresCol = 'features', labelCol = 'output',numTrees=100, maxDep

# Train model.  This also runs the indexer.
model_1 = rf_1.fit(trainingData)

# Make predictions.
predictions_1 = model_1.transform(testData)

predictions_1.show(5)

evaluator_1 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_1 = evaluator_1.evaluate(predictions_1)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_1)
```

```
22/11/28 02:16:40 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
with size 1039.5 KiB
22/11/28 02:16:45 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
with size 1355.2 KiB
22/11/28 02:16:51 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task binary
with size 1985.5 KiB

+------+-------------------+-----------------+
|output|           features|       prediction|
+------+-------------------+-----------------+
|  66.0|[0.03820530001504...|65.37242325648447|
|  66.0|[0.28265544798691...| 64.7912911071527|
|  66.0|[0.53014926553508...|66.07348658348114|
|  66.0|[1.32477892358710...|64.84387280397088|
|  66.0|[1.39127585699721...| 64.2319827191068|
+------+-------------------+-----------------+
only showing top 5 rows
```

```
[Stage 193:=====================================================> (28 + 1) / 29]
Root Mean Squared Error (RMSE) on test data = 1.30311
```

In [ ]:

In [76]:
```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator

from sklearn.metrics import roc_curve
import pyspark.sql.functions as F
import pyspark.sql.types as T
import numpy
from matplotlib import pyplot as plt

from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.sql.types import Row
from pyspark.ml.linalg import Vectors
from pyspark.ml.classification import MultilayerPerceptronClassifier
```

Linear Regression

In [ ]:

In [77]:
```
from pyspark.ml.regression import LinearRegression

lrg = LinearRegression(featuresCol = 'features', labelCol = 'output')
# maxIter: int = 100 regParam=0.0
model5 = lrg.fit(trainingData)
```

```
22/11/28 02:17:16 WARN org.apache.spark.ml.util.Instrumentation: [b579a320] regParam is zero, w
hich might cause numerical instability and overfitting.
22/11/28 02:17:18 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.g
ithub.fommil.netlib.NativeSystemBLAS
22/11/28 02:17:18 WARN com.github.fommil.netlib.BLAS: Failed to load implementation from: com.g
ithub.fommil.netlib.NativeRefBLAS
22/11/28 02:17:29 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: co
m.github.fommil.netlib.NativeSystemLAPACK
22/11/28 02:17:30 WARN com.github.fommil.netlib.LAPACK: Failed to load implementation from: co
m.github.fommil.netlib.NativeRefLAPACK
22/11/28 02:17:30 WARN org.apache.spark.ml.util.Instrumentation: [b579a320] Cholesky solver fai
led due to singular covariance matrix. Retrying with Quasi-Newton solver.
```

In [78]:
```
predictions5 = model5.transform(testData)

predictions5.show(5)

evaluator5 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse5 = evaluator5.evaluate(predictions5)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse5)
```

```
+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.03820530001504...| 66.3101328416449|
|  66.0|[0.28265544798691...| 70.3776749552288|
```

```
|  66.0|[0.53014926553508...|64.54012354110822|
|  66.0|[1.32477892358710...|66.53331446599157|
|  66.0|[1.39127585699721...|70.68842530604368|
+------+--------------------+-----------------+
only showing top 5 rows

[Stage 200:=============================================>         (24 + 4) / 29]
Root Mean Squared Error (RMSE) on test data = 1.82099
```

tune

In [79]:

```
from pyspark.ml.regression import LinearRegression

# Define LinearRegression algorithm
lrg_1 = LinearRegression(featuresCol = 'features', labelCol = 'output',regParam=2.0)
# maxIter: int = 100
model_1 = lrg_1.fit(trainingData)
predictions_1 = model_1.transform(testData)

predictions_1.show(5)

evaluator_1 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_1 = evaluator_1.evaluate(predictions_1)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_1)
```

```
+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.03820530001504...| 69.4061163237853|
|  66.0|[0.28265544798691...|70.32659323249301|
|  66.0|[0.53014926553508...|67.15931508584833|
|  66.0|[1.32477892358710...|68.02398951194853|
|  66.0|[1.39127585699721...|69.65743675854861|
+------+--------------------+-----------------+
only showing top 5 rows

[Stage 207:=====================================================>   (27 + 2) / 29]
Root Mean Squared Error (RMSE) on test data = 2.14796
```

In [80]:

```
from pyspark.ml.regression import LinearRegression

# Define LinearRegression algorithm
lrg_2 = LinearRegression(featuresCol = 'features', labelCol = 'output',maxIter=10)
model_2 = lrg_2.fit(trainingData)
predictions_2 = model_2.transform(testData)

predictions_2.show(5)

evaluator_2 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_2 = evaluator_2.evaluate(predictions_2)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_2)
```

```
22/11/28 02:18:40 WARN org.apache.spark.ml.util.Instrumentation: [edd43941] regParam is zero, w
hich might cause numerical instability and overfitting.
22/11/28 02:18:54 WARN org.apache.spark.ml.util.Instrumentation: [edd43941] Cholesky solver fai
led due to singular covariance matrix. Retrying with Quasi-Newton solver.

+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.03820530001504...|68.30209170603922|
|  66.0|[0.28265544798691...|71.10883292596793|
|  66.0|[0.53014926553508...| 66.5295163853559|
```

```
|  66.0|[1.32477892358710...| 67.3499595248948|
|  66.0|[1.39127585699721...|71.42737787287015|
+------+--------------------+-----------------+
only showing top 5 rows

[Stage 214:=============================================>         (24 + 4) / 29]
Root Mean Squared Error (RMSE) on test data = 1.87979
```

In [81]:

```
from pyspark.ml.regression import LinearRegression

# Define LinearRegression algorithm
lrg_3 = LinearRegression(featuresCol = 'features', labelCol = 'output',maxIter=30)
model_3 = lrg_3.fit(trainingData)
predictions_3 = model_3.transform(testData)

predictions_3.show(5)

evaluator_3 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_3 = evaluator_3.evaluate(predictions_3)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_3)
```

```
22/11/28 02:19:21 WARN org.apache.spark.ml.util.Instrumentation: [d938e446] regParam is zero, w
hich might cause numerical instability and overfitting.
22/11/28 02:19:34 WARN org.apache.spark.ml.util.Instrumentation: [d938e446] Cholesky solver fai
led due to singular covariance matrix. Retrying with Quasi-Newton solver.

+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.03820530001504...|66.45088998280346|
|  66.0|[0.28265544798691...|70.19812600785434|
|  66.0|[0.53014926553508...| 64.9534353872753|
|  66.0|[1.32477892358710...|66.39541368907436|
|  66.0|[1.39127585699721...|70.73252137535758|
+------+--------------------+-----------------+
only showing top 5 rows

[Stage 221:=====================================================> (28 + 1) / 29]
Root Mean Squared Error (RMSE) on test data = 1.8254
```

In [82]:

```
from pyspark.ml.regression import LinearRegression

# Define LinearRegression algorithm
lrg_4 = LinearRegression(featuresCol = 'features', labelCol = 'output',maxIter=200)
model_4 = lrg_4.fit(trainingData)
predictions_4 = model_4.transform(testData)

predictions_4.show(5)

evaluator_4 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_4 = evaluator_4.evaluate(predictions_4)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_4)
```

```
22/11/28 02:20:03 WARN org.apache.spark.ml.util.Instrumentation: [b37b5dc2] regParam is zero, w
hich might cause numerical instability and overfitting.
22/11/28 02:20:16 WARN org.apache.spark.ml.util.Instrumentation: [b37b5dc2] Cholesky solver fai
led due to singular covariance matrix. Retrying with Quasi-Newton solver.

+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.03820530001504...|66.27424177021157|
|  66.0|[0.28265544798691...|70.39226298210599|
|  66.0|[0.53014926553508...|64.66578557955714|
```

```
|  66.0|[1.32477892358710...|66.52750627777203|
|  66.0|[1.39127585699721...|70.71293591170127|
+------+--------------------+-----------------+
only showing top 5 rows

[Stage 228:=====================================================>   (27 + 2) / 29]
Root Mean Squared Error (RMSE) on test data = 1.8208
```

In [ ]:

Tensorflow

In [88]:
```
from pyspark.sql.types import *
```

In [89]:
```
to_array = udf(lambda v: v.toArray().tolist(), ArrayType(FloatType()))

df_test = testData
df_validate,df_train = trainingData.randomSplit([0.5,0.5])

df_train_pandas = df_train.withColumn('features', to_array('features')).toPandas()
df_validate_pandas = df_validate.withColumn('features', to_array('features')).toPandas()
df_test_pandas = df_test.withColumn('features', to_array('features')).toPandas()
```

In [94]:
```
import tensorflow as tf
from tensorflow import keras
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In [94], line 1
----> 1 import tensorflow as tf
      2 from tensorflow import keras

ModuleNotFoundError: No module named 'tensorflow'
```

In [90]:
```
x_train = tf.constant(np.array(df_train_pandas['features'].values.tolist()))
y_train = tf.constant(np.array(df_train_pandas['output'].values.tolist()))

x_validate = tf.constant(np.array(df_validate_pandas['features'].values.tolist()))
y_validate = tf.constant(np.array(df_validate_pandas['output'].values.tolist()))


x_test = tf.constant(np.array(df_test_pandas['features'].values.tolist()))
y_test = tf.constant(np.array(df_test_pandas['output'].values.tolist()))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In [90], line 1
----> 1 import tensorflow as tf
      2 from tensorflow import keras
      4 x_train = tf.constant(np.array(df_train_pandas['features'].values.tolist()))

ModuleNotFoundError: No module named 'tensorflow'
```

In [ ]:
```
print(x_train)
print(y_train)
```

Neural networks

In [ ]:
```
model_nn = keras.Sequential( [keras.layers.Dense(78,activation='relu'),
                              keras.layers.Dense(10,activation='relu'),
                              keras.layers.Dense(10,activation='relu'),
                              keras.layers.Dense(10,activation='relu') ,
                              keras.layers.Dense(1)] )
```

In [ ]:
```
y_pred = model_nn(x_train)
model_nn.summary()
```

In [ ]:
```
print(y_pred)
print(y_train)
```

In [ ]:
```
mse = keras.losses.MeanSquaredError()

model_nn.compile(optimizer = 'adam',
    loss=mse,
    metrics=[mse])
model_nn.fit(x_train,y_train, epochs = 20,validation_data=(x_validate,y_validate),verbose = 2)

loss = mse(y_train, y_pred).numpy()
print(loss)
```

In [ ]:
```
model_nn.evaluate(x_test,y_test, verbose = 2)
```

In [ ]:

In [ ]:

In [ ]:
```
def cross_valiation(hyper,k,s_r,x,y,logdir):

    def data_split():
        for i in range(k):
            idx=tf.range(df_train_pandas.shape[0])
            splt_idx = int(s_r*df_train_pandas.shape[0])
            idx = tf.random.shuffle(idx)
            x_train, y_train = tf.gather(x, idx[:splt_idx]), tf.gather(y, idx[:splt_idx])
            x_valid, y_valid = tf.gather(x, idx[splt_idx:]), tf.gather(y, idx[splt_idx:])
            return x_train,y_train,x_valid,y_valid

    model = keras.Sequential()
    for _ in range(hparams[HP_DEPTH]):
        model.add(keras.layers.Dense(hparams[HP_WIDTH],activation='relu'))
        model.add(keras.layers.Dense(1))
        model.compile(optimizer = 'adam',
            loss=keras.losses.MeanSquaredError(),
            metrics=[keras.losses.MeanSquaredError(name = 'MSE')])
        x_train,y_train,x_valid,y_valid = data_split()
        history = model.fit(x_train, y_train, epochs=5, verbose = 2,validation_data = (x_valid,
                  callbacks=[tf.keras.callbacks.TensorBoard(log_dir=logdir, histogram_freq=1)])
        accuracy = np.mean(history.history["MSE"])
        model.summary()
    return accuracy
```

In [ ]:
```
from tensorboard.plugins.hparams import api as hp

HP_WIDTH = hp.HParam('NN_width', hp.Discrete([10,20,30]))
HP_DEPTH = hp.HParam('NN_depth', hp.Discrete([3,4,5]))
```

```
with tf.summary.create_file_writer('logs14813/hparam_tuning').as_default():
    hp.hparams_config(
    hparams=[HP_WIDTH, HP_DEPTH],
    metrics=[hp.Metric('MSE')],
  )
```

In [ ]:

```
import datetime
for hp_width in HP_WIDTH.domain.values:
    for hp_depth in (HP_DEPTH.domain.values):
        hparams = {
            HP_WIDTH: hp_width,
            HP_DEPTH: hp_depth,
        }
        run_name = f"run-WIDTH{int(hparams[HP_WIDTH])}-DEPTH{hparams[HP_DEPTH]}"
        print('--- Starting trial: %s' % run_name)
        print({h.name: hparams[h] for h in hparams})

        run_dir = 'logs14813/hparam_tuning/' + datetime.datetime.now().strftime("%Y%m%d-%H%M%S'
        accuracy = cross_valiation(hparams,10,0.7,x_train, y_train,run_dir)

        with tf.summary.create_file_writer(run_dir).as_default():
          hp.hparams(hparams)  # record the values used in this trial
          tf.summary.scalar("MSE", accuracy, step=1)
```

Linear Regression

In [ ]:

```
tf.compat.v1.disable_eager_execution()
# tf.compat.v1.enable_eager_execution()
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np

from sklearn.datasets import load_boston

def append_bias_reshape(x_train,y_train):
    n_training_samples = x_train.shape[0]
    n_dim = x_train.shape[1]
    f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])
    l = np.reshape(y_train,[n_training_samples,1])

    return f,l

if __name__ == '__main__':


#     x,y = read_boston_data()
#     norm_features = feature_normalize(x)
    f,l = append_bias_reshape(x_train,y_train)
    n_dim = f.shape[1]

    rnd_indices = np.random.rand(len(f)) < 0.80

    x_train = f[rnd_indices]
    y_train = l[rnd_indices]
    x_test = f[~rnd_indices]
    y_test = l[~rnd_indices]



    learning_rate = 0.0001
    training_epochs = 30
    cost_history = []
    test_history = []
```

```
    X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
    Y = tf.compat.v1.placeholder(tf.float32,[None,1])
    W = tf.Variable(tf.ones([n_dim,1]))

    init = tf.compat.v1.initialize_all_variables()

    y_ = tf.matmul(X,W)
    cost = tf.reduce_mean(tf.abs(y_-Y))
    training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

    sess = tf.compat.v1.Session()
    sess.run(init)

    for epoch in range(training_epochs):
        sess.run(training_step,feed_dict={X:x_train,Y:y_train})
        c = sess.run(cost,feed_dict={X:x_train,Y:y_train})
        print(c)
        t = sess.run(cost,feed_dict={X:x_test,Y:y_test})
        print(t)
        cost_history.append(c)
        test_history.append(t)

    plt.plot(range(len(test_history)),test_history,color = 'green')
    plt.plot(range(len(cost_history)),cost_history,color = 'red')

    plt.axis([0,training_epochs,0,np.max(cost_history)])
    plt.show()
```

In [ ]:

```
# tf.compat.v1.disable_eager_execution()
# tf.compat.v1.enable_eager_execution()
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np

from sklearn.datasets import load_boston

def append_bias_reshape(x_train,y_train):
    n_training_samples = x_train.shape[0]
    n_dim = x_train.shape[1]
    f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])
    l = np.reshape(y_train,[n_training_samples,1])

    return f,l

if __name__ == '__main__':


#     x,y = read_boston_data()
#     norm_features = feature_normalize(x)
    f,l = append_bias_reshape(x_train,y_train)
    n_dim = f.shape[1]

    rnd_indices = np.random.rand(len(f)) < 0.80

    x_train = f[rnd_indices]
    y_train = l[rnd_indices]
    x_test = f[~rnd_indices]
    y_test = l[~rnd_indices]



    learning_rate = 0.001
    training_epochs = 30
    cost_history = []
    test_history = []

    X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
```

```
    Y = tf.compat.v1.placeholder(tf.float32,[None,1])
    W = tf.Variable(tf.ones([n_dim,1]))

    init = tf.compat.v1.initialize_all_variables()

    y_ = tf.matmul(X,W)
    cost = tf.reduce_mean(tf.abs(y_-Y))
    training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

    sess = tf.compat.v1.Session()
    sess.run(init)

    for epoch in range(training_epochs):
        sess.run(training_step,feed_dict={X:x_train,Y:y_train})
        c = sess.run(cost,feed_dict={X:x_train,Y:y_train})
        print(c)
        t = sess.run(cost,feed_dict={X:x_test,Y:y_test})
        print(t)
        cost_history.append(c)
        test_history.append(t)

    plt.plot(range(len(test_history)),test_history,color = 'green')
    plt.plot(range(len(cost_history)),cost_history,color = 'red')

    plt.axis([0,training_epochs,0,np.max(cost_history)])
    plt.show()
```

In [ ]:

```
# tf.compat.v1.disable_eager_execution()
# tf.compat.v1.enable_eager_execution()
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np

from sklearn.datasets import load_boston

def append_bias_reshape(x_train,y_train):
    n_training_samples = x_train.shape[0]
    n_dim = x_train.shape[1]
    f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])
    l = np.reshape(y_train,[n_training_samples,1])

    return f,l

if __name__ == '__main__':


#     x,y = read_boston_data()
#     norm_features = feature_normalize(x)
    f,l = append_bias_reshape(x_train,y_train)
    n_dim = f.shape[1]

    rnd_indices = np.random.rand(len(f)) < 0.80

    x_train = f[rnd_indices]
    y_train = l[rnd_indices]
    x_test = f[~rnd_indices]
    y_test = l[~rnd_indices]



    learning_rate = 0.00001
    training_epochs = 300
    cost_history = []
    test_history = []

    X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
    Y = tf.compat.v1.placeholder(tf.float32,[None,1])
```

```
    W = tf.Variable(tf.ones([n_dim,1]))

    init = tf.compat.v1.initialize_all_variables()

    y_ = tf.matmul(X,W)
    cost = tf.reduce_mean(tf.abs(y_-Y))
    training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

    sess = tf.compat.v1.Session()
    sess.run(init)

    for epoch in range(training_epochs):
        sess.run(training_step,feed_dict={X:x_train,Y:y_train})
        c = sess.run(cost,feed_dict={X:x_train,Y:y_train})
        print(c)
        t = sess.run(cost,feed_dict={X:x_test,Y:y_test})
        print(t)
        cost_history.append(c)
        test_history.append(t)

    plt.plot(range(len(test_history)),test_history,color = 'green')
    plt.plot(range(len(cost_history)),cost_history,color = 'red')

    plt.axis([0,training_epochs,0,np.max(cost_history)])
    plt.show()
```

In [ ]:

```
# tf.compat.v1.disable_eager_execution()
# tf.compat.v1.enable_eager_execution()
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np

from sklearn.datasets import load_boston

def append_bias_reshape(x_train,y_train):
    n_training_samples = x_train.shape[0]
    n_dim = x_train.shape[1]
    f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])
    l = np.reshape(y_train,[n_training_samples,1])

    return f,l

if __name__ == '__main__':


#     x,y = read_boston_data()
#     norm_features = feature_normalize(x)
    f,l = append_bias_reshape(x_train,y_train)
    n_dim = f.shape[1]

    rnd_indices = np.random.rand(len(f)) < 0.80

    x_train = f[rnd_indices]
    y_train = l[rnd_indices]
    x_test = f[~rnd_indices]
    y_test = l[~rnd_indices]



    learning_rate = 0.000001
    training_epochs = 3000
    cost_history = []
    test_history = []

    X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
    Y = tf.compat.v1.placeholder(tf.float32,[None,1])
    W = tf.Variable(tf.ones([n_dim,1]))
```

```
init = tf.compat.v1.initialize_all_variables()

y_ = tf.matmul(X,W)
cost = tf.reduce_mean(tf.abs(y_-Y))
training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

sess = tf.compat.v1.Session()
sess.run(init)

for epoch in range(training_epochs):
    sess.run(training_step,feed_dict={X:x_train,Y:y_train})
    c = sess.run(cost,feed_dict={X:x_train,Y:y_train})
    print(c)
    t = sess.run(cost,feed_dict={X:x_test,Y:y_test})
    print(t)
    cost_history.append(c)
    test_history.append(t)

plt.plot(range(len(test_history)),test_history,color = 'green')
plt.plot(range(len(cost_history)),cost_history,color = 'red')

plt.axis([0,training_epochs,0,np.max(cost_history)])
plt.show()
```

In [ ]: