B9DA109 Machine Learning: CA_ONE

Supervised Machine Learning – Regression

November 2022

Submitted by:
Aniket Suresh Bachewar: 10621078
Prathamesh Jadhav: 10621081
Vaibhav Tiwari: 10621051
Lecturer: Courtney Ford

Index

- **Colab code link:** https://colab.research.google.com/drive/1LiWhOcdI-TUkRL5nyFGdfyr6hlkDakSZ?usp=sharing
- **Kaggle dataset:** https://www.kaggle.com/datasets/thec03u5/fifa-18-demo-player-dataset

1. **Introduction**

Being one of the most famed sports around the world, football has proven to be well-known and versatile sports in human history.
Formerly known as soccer and originated in England during the late 19th century, the sport soon expanded to other parts of the world and professional football leagues were established. FIFA was later formed in 1904 and is now the international body that governs the sport and has since evolved the sport expeditiously to different parts of the world. Having a huge fan following from young to elderly, all-embracing it as one of the most prominent sports, the recent FIFA world cup has reached more than 3 billion football followers worldwide.

As football players are amongst the highest paid on the planet, football as a sport proves to be a lucrative business. The source of income for players ranges from club contracts, game bonuses, and media services, and would be easy to determine based on a player's level of potential. As a matter of fact, “When Mario Balotelli signed for Liverpool, a clause was put in his contract which entitled him to £1 million if he received less than 3 red cards in a match.” High-valued players (Cristiano Ronaldo **£480,000 per week**) are compensated generously with high pay, and it sometimes makes it tough for clubs, investors, and sponsors to determine the accurate value for a player based on one’s potential.

As part of the project assignment, by using a football data set that has all the key factors to judge a player's potential (both current and future predicted potential) based on his performance, abilities, and physical attributes, we can implement different regression models which will help us determine predicted potential and how the player would perform in the future, based on his level of progression.

2. **Methodology**

As part of the football dataset, we have used Supervised Machine Learning algorithms to reduce the error cost required to estimate value for an any given player. We have applied Multiple Linear Regression and compared the results with optimization algorithms such as Multi variate Gradient descent.

The models have been thoroughly pre-processed, tested and trained to find the optimum result for the provided dataset.

3. Dataset

The dataset “FIFA 18 Complete Player Dataset” is downloaded from Kaggle.com and is publicly available. The dataset consists of 75 columns in total and describes about the dependent variables, independent variables used in the project and some other additional columns.

Dataset has information of 17981 players from multiple countries and clubs. The dataset also describes the preferred playing positions of the players such as Strikers, Wing players (Right and Left), Mid fielders, Defenders and Goal keepers. We can view the physical and gaming aspects of each player as per their preferred skills and playing positions. Dataset covers player details at very granular levels like physical specifications, compensation, body conditions and gaming positions.

Overview of dataset:

Name: FIFA 18 Complete Player Dataset
Total no of columns: 75
Total no of rows: 17981
Raw dataset: **17981 rows × 75 columns**

```python
#Reading the dataset
df = pd.read_csv(r'C:/Development/Projects/Python/assignment_machine_learning/fifa_dataset_raw.csv')
df
```

[15] ✓ 0.6s Python

	Unnamed: 0	Name	Age	Photo	Nationality	Flag	Overall	Potential	Club	Club Logo	...	RB	RCB	RCM	RDM	RF	RM	RS	RW
0	0	Cristiano Ronaldo	32	https://cdn.sofifa.org/48/18/players/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	Real Madrid CF	https://cdn.sofifa.org/24/18/teams/243.png	...	61.0	53.0	82.0	62.0	91.0	89.0	92.0	91.0
1	1	L. Messi	30	https://cdn.sofifa.org/48/18/players/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	93	93	FC Barcelona	https://cdn.sofifa.org/24/18/teams/241.png	...	57.0	45.0	84.0	59.0	92.0	90.0	88.0	91.0
2	2	Neymar	25	https://cdn.sofifa.org/48/18/players/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	94	Paris Saint-Germain	https://cdn.sofifa.org/24/18/teams/73.png	...	59.0	46.0	79.0	59.0	88.0	87.0	84.0	89.0
3	3	L. Suárez	30	https://cdn.sofifa.org/48/18/players/176580.png	Uruguay	https://cdn.sofifa.org/flags/60.png	92	92	FC Barcelona	https://cdn.sofifa.org/24/18/teams/241.png	...	64.0	58.0	80.0	65.0	88.0	85.0	88.0	87.0
4	4	M. Neuer	31	https://cdn.sofifa.org/48/18/players/167495.png	Germany	https://cdn.sofifa.org/flags/21.png	92	92	FC Bayern Munich	https://cdn.sofifa.org/24/18/teams/21.png	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17976	17976	A. Kelsey	17	https://cdn.sofifa.org/48/18/players/237463.png	England	https://cdn.sofifa.org/flags/14.png	46	63	Scunthorpe United	https://cdn.sofifa.org/24/18/teams/1949.png	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17977	17977	B. Richardson	47	https://cdn.sofifa.org/48/18/players/11728.png	England	https://cdn.sofifa.org/flags/14.png	46	46	Wycombe Wanderers	https://cdn.sofifa.org/24/18/teams/1933.png	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17978	17978	J. Young	17	https://cdn.sofifa.org/48/18/players/231381.png	Scotland	https://cdn.sofifa.org/flags/42.png	46	61	Swindon Town	https://cdn.sofifa.org/24/18/teams/1934.png	...	31.0	28.0	38.0	29.0	45.0	42.0	45.0	44.0
17979	17979	J. Lundstram	18	https://cdn.sofifa.org/48/18/players/238813.png	England	https://cdn.sofifa.org/flags/14.png	46	64	Crewe Alexandra	https://cdn.sofifa.org/24/18/teams/121.png	...	47.0	46.0	45.0	47.0	43.0	45.0	41.0	44.0
17980	17980	L. Sackey	18	https://cdn.sofifa.org/48/18/players/238308.png	Ghana	https://cdn.sofifa.org/flags/117.png	46	64	Scunthorpe United	https://cdn.sofifa.org/24/18/teams/1949.png	...	40.0	45.0	30.0	38.0	29.0	30.0	31.0	29.0

17981 rows × 75 columns

Figure 1: Dataset preview

4. <u>Understanding the data</u>

- Football players can be classified into various categories (Striker, Goalkeeper, Defenders, etc.) as part of this dataset, we have used striker's attributes to determine the best predicted outcome.
- As part of data preparation, we will only apply the relevant feature to the player group and keep only suitable features applicable to the specific player type.
- There are 15 feature variables (independent variables) for determining a Striker's predicted potential.

Attributes for Striker as mentioned below:

⇒ **Striker**: 15 key features

- **Age**
- **Dribbling**
- **Acceleration**
- **Aggression**
- **Agility**
- **Balance**
- **Ball Control**
- **Composure**
- **Curve**
- **Crossing**
- **Finishing**
- **Sprint Speed**
- **Stamina**
- **Strength**
- **Vision**

The potential of a striker footballer is based on the parameters given by FIFA below:

1. **Age:**
   More mature players face stronger competitors, receive better coaching, and receive more attention.
2. **Acceleration:**
   The results show that the average time for a soccer player to travel 60 meters is 2.42 seconds
3. **Aggression:**
   An attribute that determines a player's willpower and commitment to the game.
4. **Agility:**
   How quickly and gracefully a player can control the ball.
5. **Balance:**
   A player's physical ability and overall ability to shield/hold opposing players at low speeds is determined by their balance stats.
6. **Ball Control:**
   Determines a player's ability control the ball on the field.
7. **Composure:**
   The higher the stats, the less he makes mistakes when shooting/passing. Determines player state and sense of calm and control over frustration during a match.
8. **Crossing:**
   The conversion rate of crosses after the penalty spot is only 2% but crosses taken before the penalty spot have a success rate of 5.8%.
9. **Curve:**
   The curve is used to measure a player's ability to curve the ball when passing or shooting.
10. **Dribbling:**
    Dribbling is the maneuvering of the ball by a player while moving in a certain direction, avoiding attempts by defenders to block the ball.
11. **Finishing:**
    The finish is the accuracy of the foot shot in the box.
12. **Sprint Speed:**
    Sprint Speed is a Stat cast metric intended to quantify speed more accurately by measuring how many feet per second an athlete runs in the fastest 1-second window.
13. **Stamina:**
    The player stat determines one who has enough stamina to last a full 90 minutes to affect the game.
14. **Strength:**
    The strength stat is the ability to withstand resistance.
15. **Vision:**
    Vision is the level of vision a player can correctly complete a pass.

5. Data cleaning

The process of correcting, editing, structuring, refining and eliminating garbage data within a data set is known as data cleaning.
In machine learning, we often hear a quote "Garbage in, garbage out" which means that if we choose unsatisfactory data for analysis then we end up getting unsatisfactory results.

Data cleaning process is one of the most important step to perform as it helps us to obtain best possible results.

Data cleaning process is well explained with the 1-10-100 principle :

- ✓ Cost of avoiding bad data could be $1.
- ✓ Cost of correcting bad data could be $10.
- ✓ Cost of fixing problem created due to bad data could be $100.

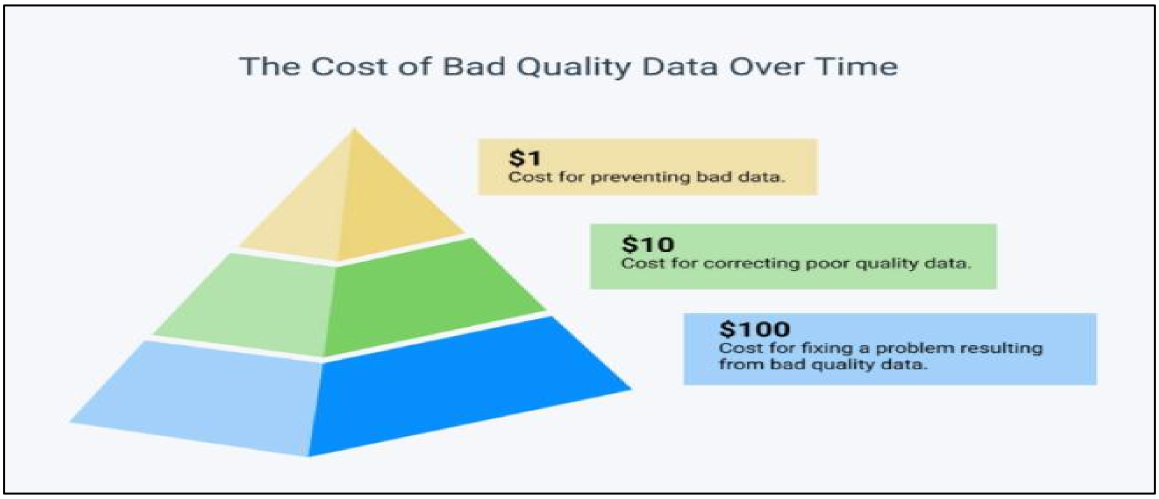

Figure 2: Cost Chart

Details about the dataset after cleaning:

Total no of columns: 20
Total no of rows: 3219 (Only striker's data)
Final dataset after cleaning: **3219 rows x 20 columns**

```
#Reading the dataset
df = pd.read_csv(r'C:/Development/Projects/Python/assignment_machine_learning/fifa_dataset_cleaned.csv')
df
```

[16] ✓ 0.2s

	Unnamed: 0	Age	Potential	Value	Wage	Acceleration	Aggression	Agility	Balance	Ball_control	Composure	Crossing	Curve	Dribbling	Finishing	Sprint_speed	Stamina	Strength	Vision	Preferred_Positions
0	0	32	94	€95.5M	€565K	89	63	89	63	93	95	85	81	91	94	91	92	80	85	ST LW
1	3	30	92	€97M	€510K	88	78	86	60	91	83	77	86	86	94	77	89	80	84	ST
2	5	28	91	€92M	€355K	79	80	78	80	89	87	62	77	85	91	83	79	84	78	ST
3	9	29	90	€77M	€275K	78	50	75	69	85	86	68	74	84	91	80	72	85	70	ST
4	13	28	89	€67.5M	€265K	88	80	90	87	87	86	80	78	90	85	84	85	72	83	RM LW ST LM
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3214	17959	18	64	€60K	€1K	69	52	54	72	45	47	34	34	43	47	68	64	58	47	CAM ST
3215	17969	18	67	€60K	€1K	62	27	69	71	46	42	32	36	51	50	65	55	41	47	ST
3216	17971	18	65	€60K	€2K	51	33	51	69	44	44	28	38	47	59	56	53	61	48	ST RM LM
3217	17978	17	61	€60K	€1K	66	26	60	77	41	50	28	32	37	47	51	33	32	37	ST
3218	17980	18	64	€50K	€1K	48	52	49	47	32	33	19	17	23	20	49	55	67	22	ST CB

3219 rows × 20 columns

Figure 3: Cleaned dataset

Validations performed on cleaned dataset:

✓ Column data types and null checks:

```
df.info()
0.7s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3219 entries, 0 to 3218
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Unnamed: 0           3219 non-null   int64
 1   Age                  3219 non-null   int64
 2   Potential            3219 non-null   int64
 3   Value                3219 non-null   object
 4   Wage                 3219 non-null   object
 5   Acceleration         3219 non-null   int64
 6   Aggression           3219 non-null   int64
 7   Agility              3219 non-null   int64
 8   Balance              3219 non-null   int64
 9   Ball_control         3219 non-null   int64
 10  Composure            3219 non-null   int64
 11  Crossing             3219 non-null   int64
 12  Curve                3219 non-null   int64
 13  Dribbling            3219 non-null   int64
 14  Finishing            3219 non-null   int64
 15  Sprint_speed         3219 non-null   int64
 16  Stamina              3219 non-null   int64
 17  Strength             3219 non-null   int64
 18  Vision               3219 non-null   int64
 19  Preferred_Positions  3219 non-null   object
dtypes: int64(17), object(3)
memory usage: 503.1+ KB
```

Figure 4

✓ Statistics check for attributes:

```
print(df.describe().T)
0.1s
Output exceeds the size limit. Open the full output data in a text editor
               count         mean          std   min     25%     50%      75%
Unnamed: 0    3219.0  8945.195713  5181.185291   0.0  4441.0  8916.0  13490.0
Age           3219.0    24.887543     4.456589  16.0    21.0    25.0     28.0
Potential     3219.0    71.337683     6.053160  52.0    67.0    71.0     75.0
Acceleration  3219.0    70.774464    11.451905  26.0    65.0    72.0     78.0
Aggression    3219.0    52.590556    15.655715  18.0    39.0    53.0     65.0
Agility       3219.0    68.349798    11.321206  29.0    61.0    69.0     76.0
Balance       3219.0    65.576266    11.993110  28.0    59.0    66.0     74.0
Ball_control  3219.0    65.008698     8.443608  31.0    60.0    65.0     71.0
Composure     3219.0    61.118360     9.886646  30.0    54.0    61.0     68.0
Crossing      3219.0    49.725070    13.529836  11.0    38.5    51.0     61.0
Curve         3219.0    53.119913    12.852460  17.0    43.0    53.0     63.0
Dribbling     3219.0    64.429015     8.908249  23.0    59.0    65.0     71.0
Finishing     3219.0    66.445480     7.679632  20.0    61.0    66.0     72.0
Sprint_speed  3219.0    71.424977    10.892728  28.0    65.0    73.0     78.0
Stamina       3219.0    65.036657    10.273901  27.0    58.0    66.0     72.0
Strength      3219.0    67.496738    12.793633  21.0    59.0    69.0     77.0
Vision        3219.0    56.309724     9.963922  22.0    49.0    56.0     64.0

                  max
Unnamed: 0    17980.0
Age              38.0
Potential        94.0
Acceleration     96.0
Aggression      106.0
...
Sprint_speed     97.0
Stamina          95.0
Strength         98.0
Vision           86.0
```

Figure 5

- ✓ Cleaning duplicate records:

```
df.drop_duplicates()
```

✓ 0.1s

	Unnamed: 0	Age	Potential	Value	Wage	Acceleration	Aggression	Agility	Balance	Ball_control	Composure	Crossing	Curve	Dribbling	Finishing	Sprint_speed	Stamina	Strength	Vision	Preferred_Positions
0	0	32	94	€95.5M	€565K	89	63	89	63	93	95	85	81	91	94	91	92	80	85	ST LW
1	3	30	92	€97M	€510K	88	78	86	60	91	83	77	86	86	94	77	89	80	84	ST
2	5	28	91	€92M	€355K	79	80	78	80	89	87	62	77	85	91	83	79	84	78	ST
3	9	29	90	€77M	€275K	78	50	75	69	85	86	68	74	84	91	80	72	85	70	ST
4	13	28	89	€67.5M	€265K	88	80	90	87	87	86	80	78	90	85	84	85	72	83	RM LW ST LM
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3214	17959	18	64	€60K	€1K	69	52	54	72	45	47	34	34	43	47	68	64	58	47	CAM ST
3215	17969	18	67	€60K	€1K	62	27	69	71	46	42	32	36	51	50	65	55	41	47	ST
3216	17971	18	65	€60K	€2K	51	33	51	69	44	44	28	38	47	59	56	53	61	48	ST RM LM
3217	17978	17	61	€60K	€1K	66	26	60	77	41	50	28	32	37	47	51	33	32	37	ST
3218	17980	18	64	€50K	€1K	48	52	49	47	32	33	19	17	23	20	49	55	67	22	ST CB

3219 rows × 20 columns

Figure 6

- ✓ Format column names:

```
#formatting column names
dfold = pd.read_csv(r'C:/Development/Projects/Python/assignment_machine_learning/fifa_dataset_raw.csv')
print(dfold.columns)
df = pd.read_csv(r'C:/Development/Projects/Python/assignment_machine_learning/fifa_dataset_cleaned.csv')

formatted_columns_name_arr = []
for column in df.columns :
    formatted_column = column.replace(" ", "_")
    formatted_columns_name_arr.append(formatted_column)

df.columns = formatted_columns_name_arr

print(df.columns)
```

✓ 0.3s

```
Index(['Unnamed: 0', 'Name', 'Age', 'Photo', 'Nationality', 'Flag', 'Overall',
       'Potential', 'Club', 'Club Logo', 'Value', 'Wage', 'Special',
       'Acceleration', 'Aggression', 'Agility', 'Balance', 'Ball control',
       'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishing',
       'Free kick accuracy', 'GK diving', 'GK handling', 'GK kicking',
       'GK positioning', 'GK reflexes', 'Heading accuracy', 'Interceptions',
       'Jumping', 'Long passing', 'Long shots', 'Marking', 'Penalties',
       'Positioning', 'Reactions', 'Short passing', 'Shot power',
       'Sliding tackle', 'Sprint speed', 'Stamina', 'Standing tackle',
       'Strength', 'Vision', 'Volleys', 'CAM', 'CB', 'CDM', 'CF', 'CM', 'ID',
       'LAM', 'LB', 'LCB', 'LCM', 'LDM', 'LF', 'LM', 'LS', 'LW', 'LWB',
       'Preferred Positions', 'RAM', 'RB', 'RCB', 'RCM', 'RDM', 'RF', 'RM',
       'RS', 'RW', 'RWB', 'ST'],
      dtype='object')
Index(['Unnamed:_0', 'Age', 'Potential', 'Value', 'Wage', 'Acceleration',
       'Aggression', 'Agility', 'Balance', 'Ball_control', 'Composure',
       'Crossing', 'Curve', 'Dribbling', 'Finishing', 'Sprint_speed',
       'Stamina', 'Strength', 'Vision', 'Preferred_Positions'],
      dtype='object')
```

Figure 7

✓ Dropping columns:

Dropping columns which contributes less to predictions.

```
df.drop(['Value','Wage', 'Preferred_Positions'], axis=1, inplace=True)
print(df)
```

✓ 0.1s

```
Output exceeds the size limit. Open the full output data in a text editor
      Unnamed: 0  Age  Potential  Acceleration  Aggression  Agility  Balance
0              0   32         94            89          63       89       63
1              3   30         92            88          78       86       60
2              5   28         91            79          80       78       80
3              9   29         90            78          50       75       69
4             13   28         89            88          80       90       87
...          ...  ...        ...           ...         ...      ...      ...
3214       17959   18         64            69          52       54       72
3215       17969   18         67            62          27       69       71
3216       17971   18         65            51          33       51       69
3217       17978   17         61            66          26       60       77
3218       17980   18         64            48          52       49       47

      Ball_control  Composure  Crossing  Curve  Dribbling  Finishing  \
0               93         95        85     81         91         94
1               91         83        77     86         86         94
2               89         87        62     77         85         91
3               85         86        68     74         84         91
4               87         86        80     78         90         85
...            ...        ...       ...    ...        ...        ...
3214            45         47        34     34         43         47
3215            46         42        32     36         51         50
3216            44         44        28     38         47         59
3217            41         50        28     32         37         47
3218            32         33        19     17         23         20
...
3217            51       33        32       37
3218            49       55        67       22

[3219 rows x 17 columns]
```

Figure 8

6. Data preparation

The process of converting the raw data into productive data which would help to obtain conclusive understanding of data and would help data scientists and analysts in making predictions by running through the machine learning model and optimization algorithms is known as Data Preparation.

During dataset analysis we found some incorrect values in the dataset, so we noted all the patterns of the incorrect data and transformed the data by running python code on dataset to create a clean dataset.

- ✓ Converting data containing symbols & operators into appropriate column values.

```python
striker_data_columns_arr = ['Age', 'Potential', 'Acceleration', 'Aggression', 'Agility', 'Balance', 'Ball_control', 'Composure',
'Crossing', 'Curve', 'Dribbling', 'Finishing', 'Sprint_speed', 'Stamina', 'Strength', 'Vision']
for column in striker_data_columns_arr :
    entire_column_dataload_arr = []
    for row_data in df[column] :
        if str(row_data).__contains__("+") :
            format_row_data = str(row_data).split("+")
            format_row_data = int(format_row_data[0]) + int(format_row_data[1])
            entire_column_dataload_arr.append(format_row_data)
        elif str(row_data).__contains__("-") :
            format_row_data = str(row_data).split("-")
            format_row_data = int(format_row_data[0]) - int(format_row_data[1])
            entire_column_dataload_arr.append(format_row_data)
        elif str(row_data).__contains__("/"):
            entire_column_dataload_arr.append(0)
        else :
            entire_column_dataload_arr.append(row_data)
    df[column] = entire_column_dataload_arr
    print(df.head)
```

✓ 0.2s

```
Output exceeds the size limit. Open the full output data in a text editor
<bound method NDFrame.head of       Unnamed: 0  Age  Potential   Value   Wage  Acceleration  Aggression  \
0              0   32         94  €95.5M  €565K            89          63
1              3   30         92    €97M  €510K            88          78
2              5   28         91    €92M  €355K            79          80
3              9   29         90    €77M  €275K            78          50
4             13   28         89  €67.5M  €265K            88          80
...          ...  ...        ...     ...    ...           ...         ...
3214       17959   18         64    €60K    €1K            69          52
3215       17969   18         67    €60K    €1K            62          27
3216       17971   18         65    €60K    €2K            51          33
3217       17978   17         61    €60K    €1K            66          26
3218       17980   18         64    €50K    €1K            48          52

      Agility  Balance  Ball_control  Composure  Crossing  Curve  Dribbling  \
0          89       63            93         95        85     81         91
1          86       60            91         83        77     86         86
2          78       80            89         87        62     77         85
3          75       69            85         86        68     74         84
4          90       87            87         86        80     78         90
...       ...      ...           ...        ...       ...    ...        ...
3214       54       72            45         47        34     34         43
3215       69       71            46         42        32     36         51
3216       51       69            44         44        28     38         47
3217       60       77            41         50        28     32         37
3218       49       47            32         33        19     17         23
...
3217         47          51      33        32     37              ST
3218         20          49      55        67     22           ST CB

[3219 rows x 20 columns]>
```

Figure 9

✓ Converting amount values containing symbols “€” and “K” into numerical number.

```
dfraw = pd.read_csv(r'C:/Development/Projects/Python/assignment_machine_learning/fifa_dataset_raw.csv')
df = pd.read_csv(r'C:/Development/Projects/Python/assignment_machine_learning/fifa_dataset_cleaned.csv')
formatted_wage_arr = []
for wage in df.Wage :
    wage_formatting = wage.replace("€", "")
    wage_formatting = wage_formatting.replace("K", "000")
    formatted_wage_arr.append(wage_formatting)

df["Wage"] = formatted_wage_arr
print("Wage column from raw dataset and cleaned dataset ")
print(dfraw["Wage"])
print(df["Wage"])
```

✓ 0.6s

```
Wage column from raw dataset and cleaned dataset
0        €565K
1        €565K
2        €280K
3        €510K
4        €230K
         ...
17976      €1K
17977      €1K
17978      €1K
17979      €1K
17980      €1K
Name: Wage, Length: 17981, dtype: object
0       565000
1       510000
2       355000
3       275000
4       265000
         ...
3214      1000
3215      1000
3216      2000
3217      1000
3218      1000
Name: Wage, Length: 3219, dtype: object
```

Figure 10

7. **Preliminary data analysis (Exploratory)**

```
##############################################################################################
######################### C L A S S I F Y I N G V A R I A B L E S #####################
##############################################################################################
Classifying variables in data set...
Printing upto 30 columns max in each category:
    Numeric Columns : ['Potential', 'Wage']
    Integer-Categorical Columns: ['Age', 'Acceleration', 'Aggression', 'Agility', 'Balance', 'Ball_control', 'Composure', 'Crossing', 
'Curve', 'Dribbling', 'Finishing', 'Sprint_speed', 'Stamina', 'Strength', 'Vision']
    Discrete String Columns: ['Value', 'Preferred_Positions']
    20 Predictors classified...
        This does not include Target column(s)
        1 variables removed since they were ID or low-information variables
   Categorical variables %s
("   ['Age', 'Acceleration', 'Aggression', 'Agility', 'Balance', "
 "'Ball_control', 'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishing', "
 "'Sprint_speed', 'Stamina', 'Strength', 'Vision']")
   Continuous variables %s
"   ['Potential', 'Wage']"
   Discrete string variables %s
"   ['Value', 'Preferred_Positions']"'
Number of All Scatter Plots = 3
```

Below are the pair wise scatter plots for the continuous variables of the dataset. Attached scatter plot reference below:

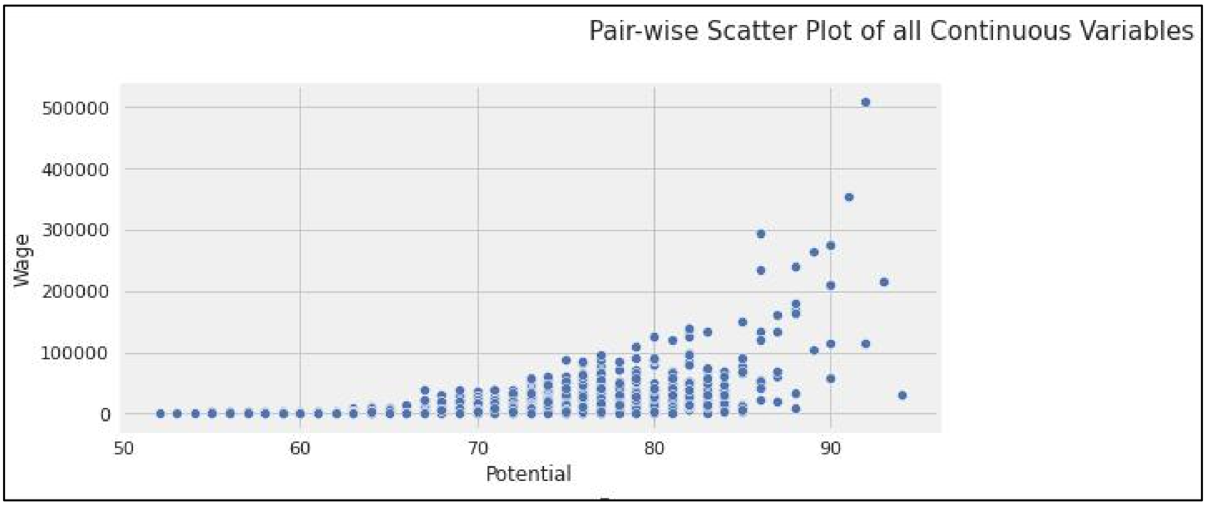

Figure 11: Scatter plot

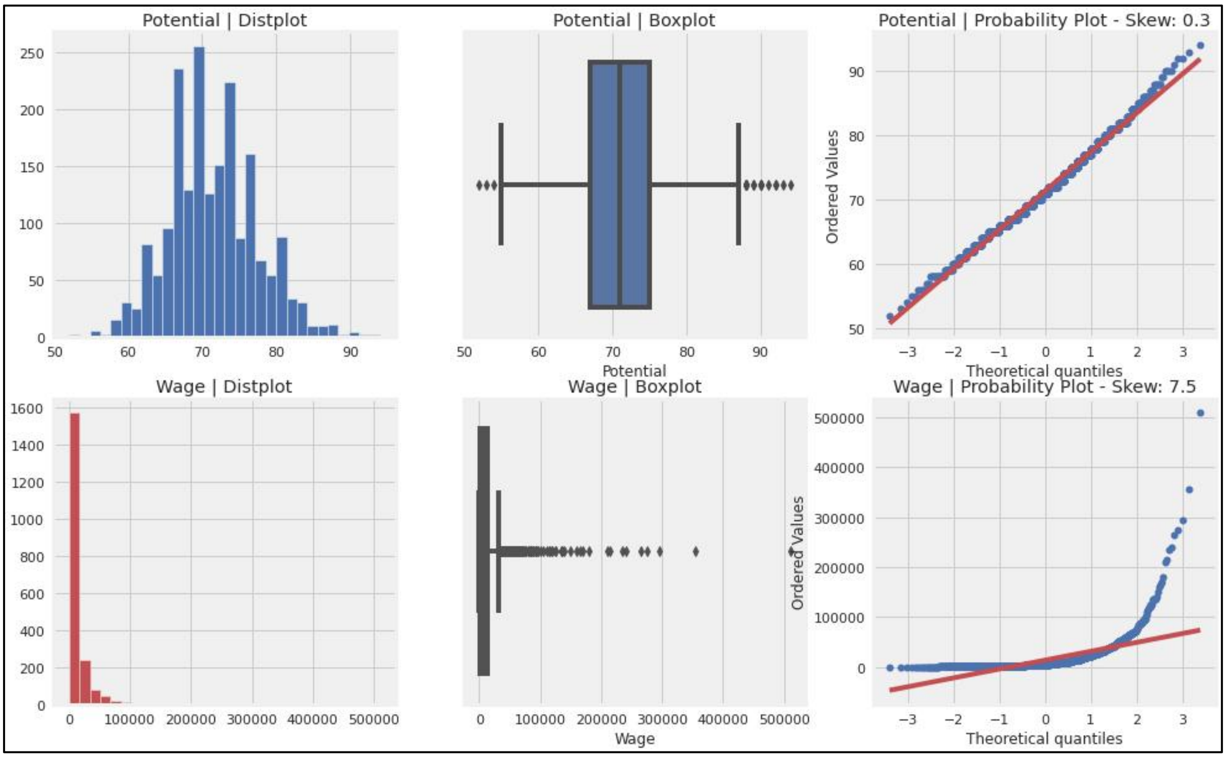

Figure 12: AutoViz generated plots

Overview

Dataset Statistics

Number of Variables	20
Number of Rows	3219
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	960.2 KB
Average Row Size in Memory	305.4 B
Variable Types	Numerical: 18 Categorical: 2

Dataset Insights

`Acceleration` and `Sprint_speed` have similar distributions	Similar Distribution
`Agility` and `Strength` have similar distributions	Similar Distribution
`Ball_control` and `Dribbling` have similar distributions	Similar Distribution
`Dribbling` and `Stamina` have similar distributions	Similar Distribution
`Wage` is skewed	Skewed
`Value` has a high cardinality: 162 distinct values	High Cardinality
`Preferred_Positions` has a high cardinality: 208 distinct values	High Cardinality

Figure 13: AutoViz generated overview

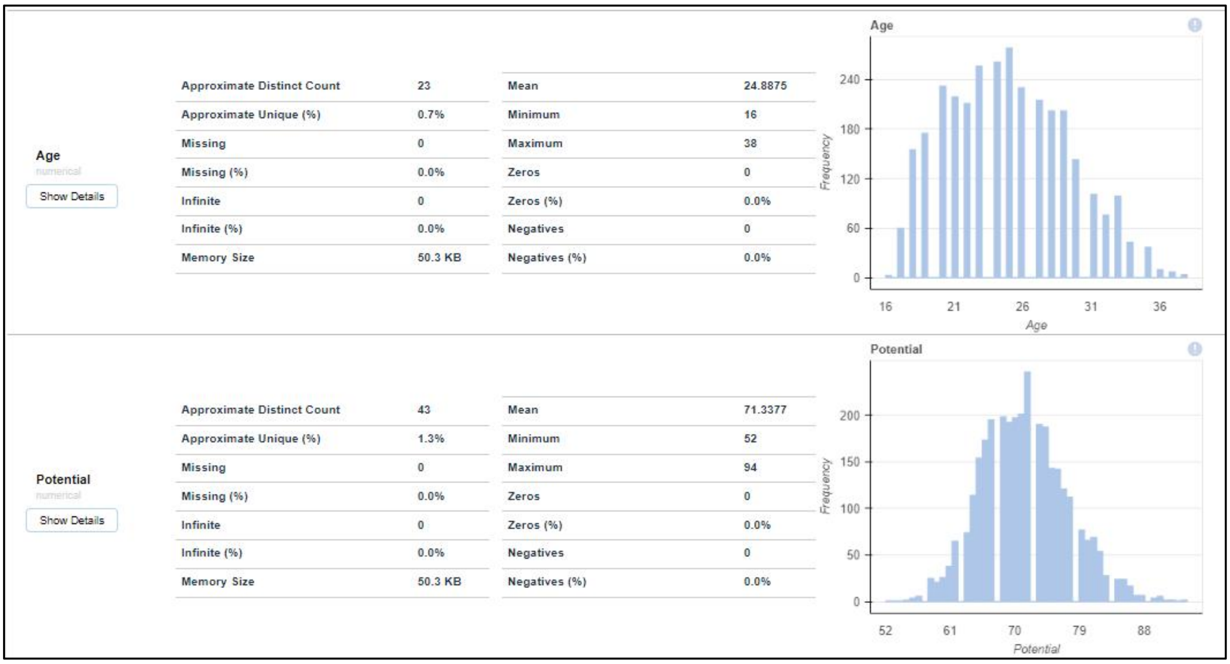

Figure 14

Figure 15

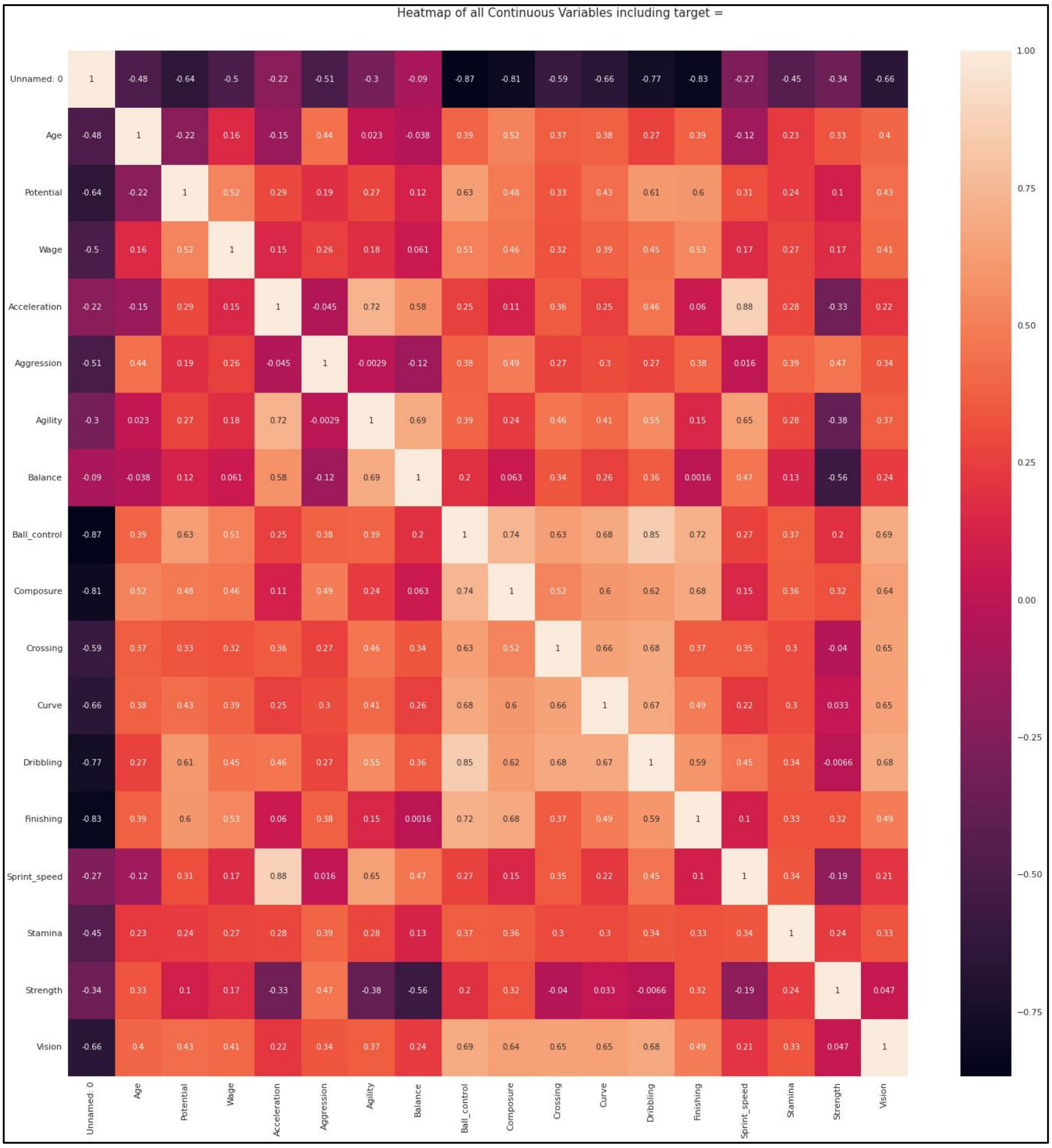

Figure 16: AutoViz heatmap

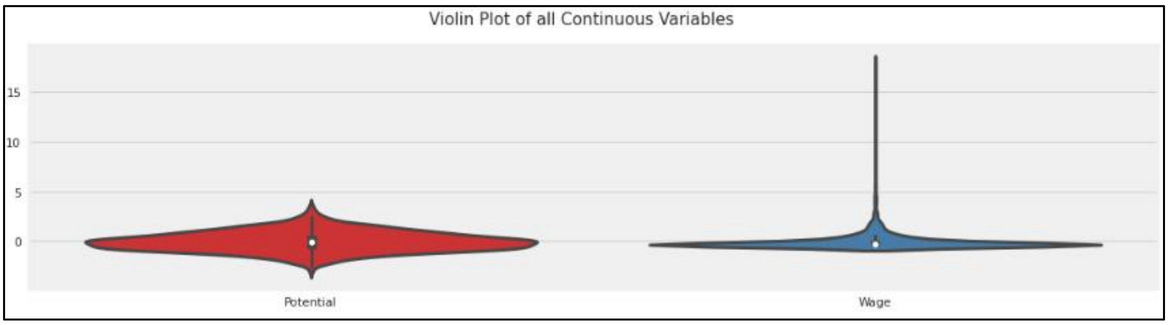

Figure 17: AutoViz Violin plot

8. Feature selection & Model development

- Data used to create a predictive model helps us yield practical use of these informative results.
- Haven choosing the right set of features can help us remove irrelevant features and noisy data from our machine learning models thus achieving higher and more precise prediction power.
- Reduced model training time and increased model performance play a crucial role in regression modeling which is achieved by the efficient selection of feature variables.

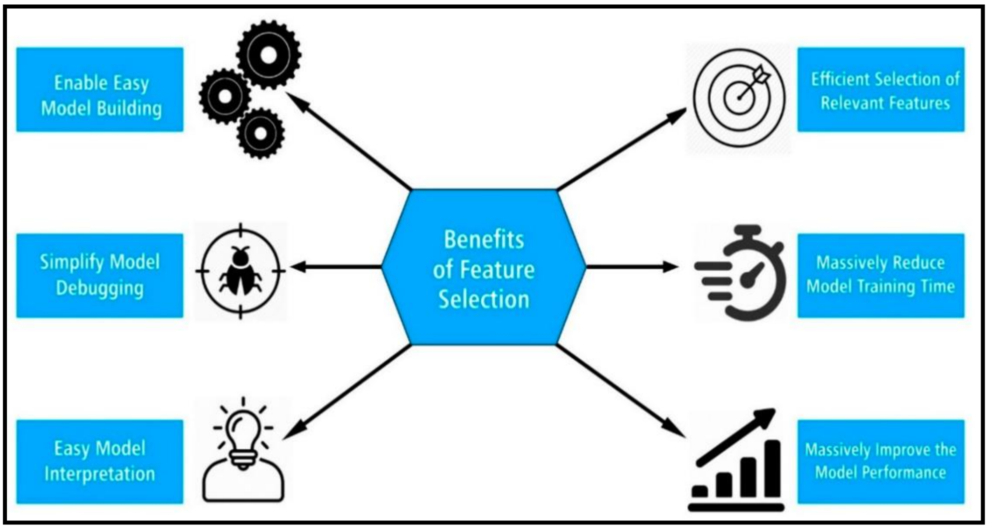

Figure 18: Feature selection benefits

- Features used in the model:

```
file_name = "/content/fifa_dataset_cleaned_potential.csv"
model = pd.read_csv(file_name,low_memory=False)
# Selecting features for model training
model_features = model.loc[:, ['Age', 'Potential', 'Value', 'Wage', 'Acceleration', 'Aggression',
                               'Agility', 'Balance', 'Ball_control','Composure', 'Crossing', 'Curve',
                               'Dribbling', 'Finishing', 'Sprint_speed', 'Stamina', 'Strength', 'Vision',
                               'Preferred_Positions']]
```

- There are 15 feature variables (independent variables) for determining a Striker's predicted potential.

Player types as mentioned below:

- **Striker**: 15 key features
  Age, potential, acceleration, aggression, balance, ball control, composure, dribbling, finishing, positioning, sprint speed, stamina, strength, vision

Using key features with machine learning regression models will now help us derive meaningful insights from the data and improve decision-making capabilities which were earlier limited due to manual intervention.

9. <u>Model Evaluation</u>

- As part of model evaluation, we will verify and determine players potential based on striker's abilities which will quantify the quality of our system's predictions.
- Below are parameters evaluate the quality of models generated by our **machine learning algorithms.**

Model training results based on Multiple Linear Regression:

R^2 score:

- R-squared shows how well the data fit the regression model which is the **goodness of fit**.
- **R^2 score** on test data for multiple linear regression model is: **0.7977**

Formula

$$R^2 = 1 - \frac{RSS}{TSS}$$

R^2 = coefficient of determination
RSS = sum of squares of residuals
TSS = total sum of squares

Mean Square Error (MSE):

- **Mean Square Error (MSE)** is the mean of the squared errors.
- **MSE** on test data for multiple linear regression model is: **7.3114**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Mean Absolute Error (MAE):

- **Mean Absolute Error (MAE)** is the mean of the absolute value of the errors.
- **MAE** on test data for multiple linear regression model is: **2.0903**

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Root Mean Square Error (RMSE):

- **Root Mean Square Error (RMSE)** is the square of the mean of the squared errors.
- **RMSE** on test data for multiple linear regression model is: **2.7039**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Model training results based on Gradient Descent Algorithm:

Coefficients of Regression on test data for gradient descent algorithm:

[-0.05750458829931079, 0.0022839191139411856, 0.004979059102198265, -0.0023008861178433994, 0.00019391094998104388, 0.025233486202504374, 0.011721531720040616, 9.07194273427946e-05, 0.0044043401288903425, 0.00763850576681522, 0.032870409350503185, 0.0038869638238184377, -0.0026933134468111997, 0.006774466292053565, 0.008289468306520975]

Intercept: **4.2638469800700864**

The **Cost Function** test data for gradient descent algorithm:

0.0418457475698129

10. Model comparison

In the project we have compared Multilinear Regression with Gradient Descent to compute Mean Square Error MSE and Cost function.

Multilinear Regression is a model or a technique to perform the data analysis on the dataset wherein the Gradient Descent is an optimization algorithm which is applied on the dataset to minimize the cost function.

Note: Numerical comparison metrics computed based on the selected dataset can be found in the section Model Evaluation.

11. **Conclusion**

The purpose of this report was to prepare and test a model for predicting the Potential of a player in the market with the help of a machine learning models/techniques and algorithms. Based on the analysis performed on our selected dataset (performance, abilities and physical attributes) we have derived that the Gradient Descent is better than the Multilinear regression. This is concluded based on the values of Cost Function and MSE (Mean Square Error) calculated by the approach stated previously in the report. In summary, the derived cost function from the Gradient descent seems to be significantly lower than the MSE derived through the Multilinear regression.
It can be concurred that this approach can also be stretched out to the different areas, provided a better player game performative framework is created with different associations and independent research agencies. We can implement different regression models which will help us determine predicted potential and how the player would perform in the future based on his level of progression. Regardless, this work will proceed with the general data agreement that, not all footballers (by their exchanges) are assessed every year, or footballer’s player game performance isn't completely assessed.

12. **References**

- Kaggle dataset:
  - https://www.kaggle.com/datasets/thec03u5/fifa-18-demo-player-dataset

- Collab code link:
  - https://colab.research.google.com/drive/1LiWhOcdI-TUkRL5nyFGdfyr6hlkDakSZ?usp=sharing

  - Introduction references:
    - https://www.footballhistory.org/world-cup/index.html
    - https://www.thisisanfield.com/2016/12/mario-balotellis-liverpool-contract-included-incredible-1-million-behaviour-clause/
    - https://en.wikipedia.org/wiki/FIFA
    - https://footballiconic.com/how-footballers-get-paid/

  - Data cleaning references:
    - https://monkeylearn.com/blog/data-cleaning-steps/

  - Feature selection references:
    - https://www.heavy.ai/technical-glossary/feature-selection
    - https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/
    - https://en.wikipedia.org/wiki/Feature_selection

  - Other references:
    - https://github.com/PhongHoangg/Gradient-Descent-for-Multivariate-Regression/blob/main/Gradient%20Descent%20for%20Multivariate%20Regression.ipynb