Machine Learning mini project

Nicolás Muñoz

Project Overview.

- **Dataset description**: This dataset keeps detailed information on approximately 17.000 fifa football players. It emcompases a wide array of players data points, including players name, nationality, positions, player ratings, potential, and various skill attributes.

- **The aim of this project is to predict the value in € of the football players.**

Data Selection and Preparation.

1- For the Data Analysis process, the dataset is imported in Python, using Pandas library.

2- To further comprehend and organize the Data, the following methods and functions are applied : shape, info(), dtypes, mapping and lambda.

3- A third step was to drop null values, drop columns with no relevance for the ML objective.

Final Data Frame.

football_players

	age	height_cm	weight_kgs	positions	potential	value_euro	preferred_foot	international_reputation(1-5)	weak_foot(1-5)	skill_moves(1-5)	national_rating
0	31	170.18	72.1	attack	94	110500000.0	1	5	4	4	82.0
1	27	154.94	76.2	midfield	89	69500000.0	0	3	5	4	78.0
2	25	190.50	83.9	midfield	91	73000000.0	0	4	4	5	84.0
3	27	162.56	59.0	attack	88	62000000.0	0	3	4	4	83.0
5	27	193.04	92.1	defence	90	59500000.0	0	3	3	2	81.0
...	...	...	...	...	...	...	...	...	...	...	...
17940	28	172.72	76.2	attack	91	93000000.0	0	4	4	4	85.0
17941	27	154.94	69.9	midfield	92	102000000.0	0	4	5	4	85.0
17942	28	193.04	76.2	GK	93	72000000.0	0	4	3	1	85.0
17943	27	175.26	68.0	attack	92	108000000.0	0	5	5	5	81.0
17944	34	187.96	83.0	attack	94	77000000.0	0	5	4	5	82.0

857 rows × 40 columns

Final Data Frame.

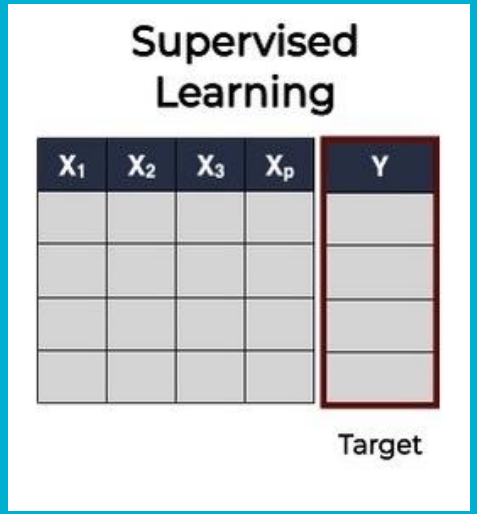
football_players

crossing	finishing	heading_accuracy	short_passing	volleys	dribbling	curve	freekick_accuracy	long_passing	ball_control	acceleration	sprint_speed	agility
86	95	70	92	86	97	93	94	89	96	91	86	93
88	81	52	91	80	84	86	87	89	91	76	73	80
80	75	75	86	85	87	85	82	90	90	71	79	76
86	77	56	85	74	90	87	77	78	93	94	86	94
53	52	83	79	45	70	60	70	81	76	74	77	61
...	...	...	...	...	...	...	...	...	...	...	...	...
81	84	61	89	80	95	83	79	83	94	94	88	95
93	82	55	92	82	86	85	83	91	91	78	76	79
17	13	21	50	13	18	21	19	51	38	57	58	62
83	87	62	84	84	96	88	87	80	95	94	90	96
84	94	89	81	87	88	81	76	77	94	89	91	87

Featuring Engineering and Selection.

Type of Machine Learning:

- The type of ML use for this Data is Supervised Learning.



Featuring Engineering and Selection.

football_players

	age	height_cm	weight_kgs	positions	potential	value_euro
0	31	170.18	72.1	attack	94	110500000.0
1	27	154.94	76.2	midfield	89	69500000.0
2	25	190.50	83.9	midfield	91	73000000.0
3	27	162.56	59.0	attack	88	62000000.0
5	27	193.04	92.1	defence	90	59500000.0
...	...	...	...	...	...	...
17940	28	172.72	76.2	attack	91	93000000.0
17941	27	154.94	69.9	midfield	92	102000000.0
17942	28	193.04	76.2	GK	93	72000000.0
17943	27	175.26	68.0	attack	92	108000000.0
17944	34	187.96	83.0	attack	94	77000000.0

857 rows × 40 columns

I want to predict the value in € for football players, that means the type of problems is a **Regression** problem. Too, that the variable to be predicted is a **Continuous/Numerical** with an infinite number of possible outcomes.

Featuring Engineering and Selection.

- Another important step for this type of ML model is to transform all the categorical values into numerical values, using one hot encoding technique.

```
categorical_num= pd.get_dummies(categorical, drop_first = True)
categorical_num
```

	positions_attack	positions_defence	positions_midfield
0	True	False	False
1	False	False	True
2	False	False	True
3	True	False	False
5	False	True	False
...	...	...	...
17940	True	False	False
17941	False	False	True
17942	False	False	False
17943	True	False	False
17944	True	False	False

857 rows x 3 columns

Featuring Engineering and Selection.

Ready to define Features and Target.

```
features = pd.concat([categorical_num, numerical], axis = 1).drop(columns=['value_euro'])
features
```

	positions_attack	positions_defence	positions_midfield	age	height_cm	weight_kgs	potential
0	True	False	False	31	170.18	72.1	94
1	False	False	True	27	154.94	76.2	89
2	False	False	True	25	190.50	83.9	91
3	True	False	False	27	162.56	59.0	88
5	False	True	False	27	193.04	92.1	90
...	...	...	...	...	...	...	...
17940	True	False	False	28	172.72	76.2	91
17941	False	False	True	27	154.94	69.9	92
17942	False	False	False	28	193.04	76.2	93
17943	True	False	False	27	175.26	68.0	92
17944	True	False	False	34	187.96	83.0	94

857 rows × 41 columns

```
target =football_players["value_euro"]
target
```

```
0        110500000.0
1         69500000.0
2         73000000.0
3         62000000.0
5         59500000.0
             ...
17940     93000000.0
17941    102000000.0
17942     72000000.0
17943    108000000.0
17944     77000000.0
Name: value_euro, Length: 857, dtype: float64
```

Train Test Split and Feature Scaling.

PERFORM TRAIN TEST SPLIT

```
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = 0.20, random_state=0)
```

FEATURE SCALING

```
normalizer = MinMaxScaler()
normalizer.fit(X_train)
```

```
▾ MinMaxScaler
MinMaxScaler()
```

```
X_train_norm = normalizer.transform(X_train)
X_test_norm = normalizer.transform(X_test)
```

```
X_train_norm = pd.DataFrame(X_train_norm, columns = X_train.columns)
X_train_norm.head()
```

	positions_attack	positions_defence	positions_midfield	age	height_cm	weight_kgs	potential	preferred_foot	international_reputation(1-5)	weak_foot(1-5)
0	0.0	0.0	1.0	0.619048	0.00	0.343137	0.25000	0.0	0.00	1.00
1	0.0	0.0	1.0	0.476190	0.00	0.365196	0.50000	1.0	0.25	0.75
2	0.0	1.0	0.0	0.571429	0.45	0.387255	0.40625	0.0	0.25	0.50
3	1.0	0.0	0.0	0.666667	0.60	0.666667	0.90625	0.0	1.00	0.75
4	0.0	0.0	1.0	0.571429	0.00	0.267157	0.31250	0.0	0.00	0.25

Model Selection: KNN Regressor and Linear Regression.

KNN REGRESSOR

```
knn = KNeighborsRegressor(n_neighbors=3)
```

```
knn.fit(X_train_norm, y_train)
```

▾ KNeighborsRegressor

```
KNeighborsRegressor(n_neighbors=3)
```

```
knn.score(X_test_norm, y_test)
```

```
0.6269702696346141
```

LINEAR REGRESSION

```
lin_reg = LinearRegression()
```

```
lin_reg.fit(X_train_norm, y_train)
```

▾ LinearRegression

```
LinearRegression()
```

```
pred = lin_reg.predict(X_test_norm)
print("MAE", mean_absolute_error(pred, y_test))
print("RMSE", mean_squared_error(pred, y_test, squared=False))
print("R2 score", lin_reg.score(X_test_norm, y_test))
```

```
MAE 5988153.292849529
RMSE 9694806.413702432
R2 score 0.7100182652929481
```

Model Selection: Decision Tree.

```
DECISION TREE
```

```
tree = DecisionTreeRegressor(max_depth=7)
```

TRAINING THE MODEL

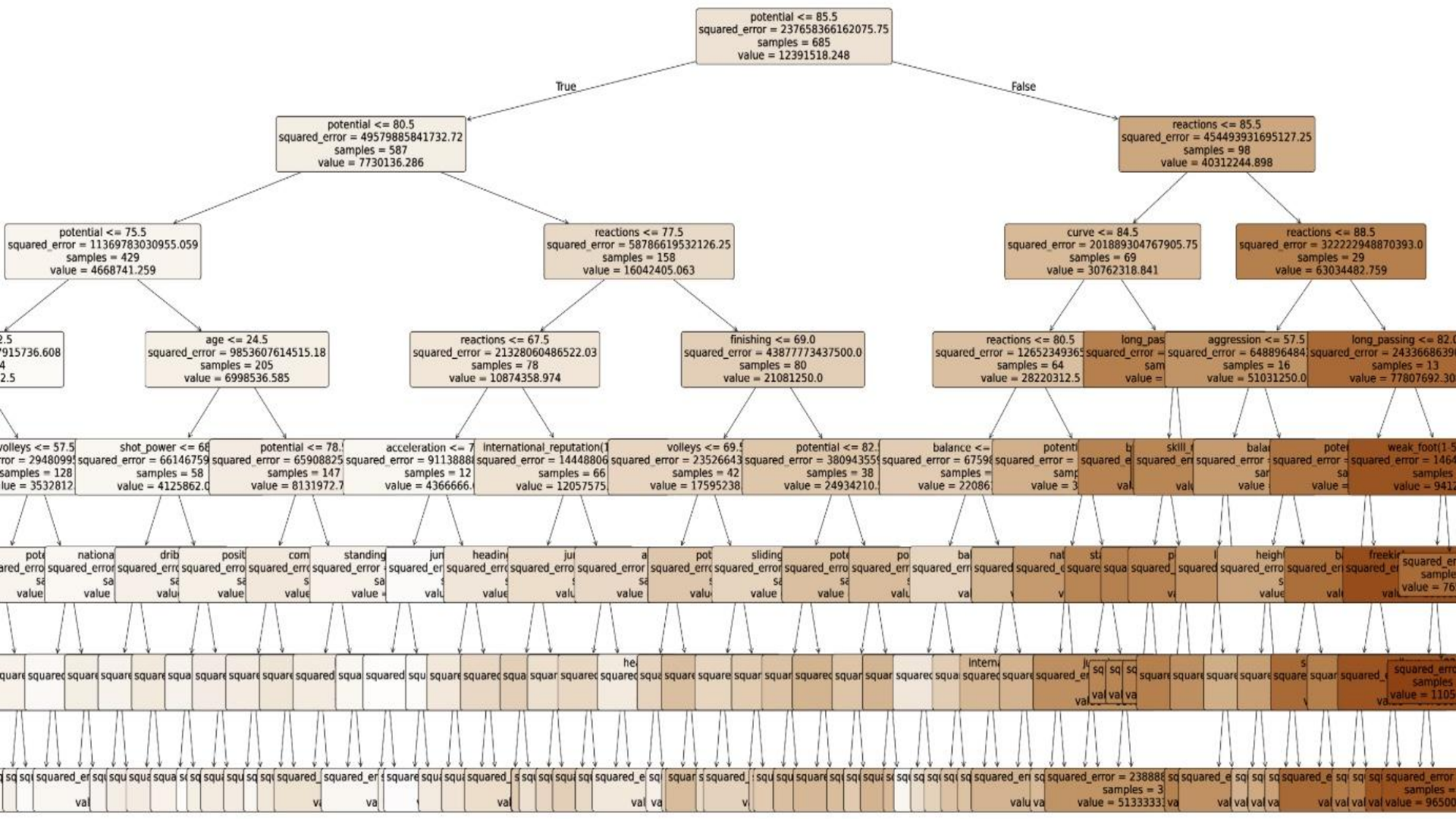
```
tree.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=7)
```

```
pred = tree.predict(X_test)

print("MAE", mean_absolute_error(pred, y_test))
print("RMSE", mean_squared_error(pred, y_test, squared=False))
print("R2 score", tree.score(X_test, y_test))
```

```
MAE 3696578.425843131
RMSE 7287933.704541087
R2 score 0.8361293458702922
```


potential <= 85.5
squared_error = 2376583661620 75.75
samples = 685
value = 12391518.248
True
False
potential <= 80.5
squared_error = 49579885841732.72
samples = 587
value = 7730136.286
reactions <= 85.5
squared_error = 454493931695127.25
samples = 98
value = 40312244.898
potential <= 75.5
squared_error = 11369783030955.059
samples = 429
value = 4668741.259
reactions <= 77.5
squared_error = 58786619532126.25
samples = 158
value = 16042405.063
curve <= 84.5
squared_error = 201889304767905.75
samples = 69
value = 30762318.841
reactions <= 88.5
squared_error = 322222948870393.0
samples = 29
value = 63034482.759
age <= 24.5
squared_error = 9853607614515.18
samples = 205
value = 6998536.585
reactions <= 67.5
squared_error = 21328060486522.03
samples = 78
value = 10874358.974
finishing <= 69.0
squared_error = 43877773437500.0
samples = 80
value = 21081250.0
reactions <= 80.5
samples = 64
value = 28220312.5
aggression <= 57.5
samples = 16
value = 51031250.0
samples = 13

Assemble Methods

RANDOM FOREST

```
forest = RandomForestRegressor(n_estimators=100,
                               max_depth=20)
```

Training the model

```
forest.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(max_depth=20)
```

```
pred = forest.predict(X_test)

print("MAE", mean_absolute_error(pred, y_test))
print("RMSE", mean_squared_error(pred, y_test, squared=False))
print("R2 score", forest.score(X_test, y_test))
```

```
MAE 2292868.3139534886
RMSE 4718928.643106978
R2 score 0.9312964495488657
```

R2 measures how well the observed outcomes are replicated by the model.

- We are now R2 = 0.93

Machine Learning mini Project.

Thank you!