Buying a soccer team

1. ABSTRACT

As we are progressing into a world where sports have become a vital part of our lives, it has also become the hot market for investors to gain better returns and interact with the audience and make their presence felt. Also, we can see that there has been a surge in sports viewership which leads to more tournaments and capitalizing on them has become a difficult task for an investor. We have taken up a challenge to help major investors to pick the best players amongst 18000 Soccer players to build a dream Soccer team which can participate and outperform other clubs in major leagues. We have leveraged Machine learning algorithms to classify potential team members in our club and potential budget for an investor to optimize their market gains. As a result, we have come up with a strategy to build the best team whilst keeping in mind the investor's budget which has a limit of 1 billion Euros.

2. INTRODUCTION

We have a FIFA dataset in which there are few columns named - rating, release clause and wages. We assume that we do not have these variables in upcoming out of time datasets. These can be used in various formations like rating a player as a better performing player or moderate or not up to the mark player. Additionally, it can also translate into a player who should be sent an invite to some club gatherings, events etc. We built 2 models using Supervised Learning on Rating variables and we made it a classification problem by splitting this variable into 2 classes: greater than or equal to 70 and less than 70 as our potential club member or not. We chose 70 as our threshold as most of the major clubs have only players whose rating is greater than 70. In order to compete with them, we are inclined to have only those players having ratings greater than our threshold. Additionally, we have worked with our model results to predict the cost to investors for offering a club membership annually. Our second model utilizes predicted rating class obtained from our previous best classifiers instead of “actual rating" and here, and a combination of *release_clause and annual wages* as cost to investors as our dependent variable.

3. DATASET

We are using the FIFA 2019 and 2020 data from the Kaggle FIFA complete player dataset. FIFA complete player dataset contains 18k+ unique players and 100+ attributes extracted from the latest edition of FIFA. It contains:

- Files present in CSV format.
- FIFA 2020 - 18,278 unique players and 104 attributes for each player. (Test dataset)
- FIFA 2019 - 17,770 unique players and 104 attributes for each player. (Train dataset)
- URL of the scraped player.
- Player positions, with the role in the club and in the national team.
- Player attributes with statistics as Attacking, Skills, Defense, Mentality, GK Skills, etc.
- Player personal data like Nationality, Club, DateOfBirth, Wage, Salary, etc.

4. DATA CLEANING

- At some places, both the datasets have different data types of the same features. After reading the data dictionary, we brought them into sync.
- Some variables have in-built formulas, so we corrected their formatting.
- We removed *'sofifa_id', 'player_url', 'short_name', 'long_name', 'real_face', 'dob', 'gk_diving', 'gk_handling', 'gk_kicking', 'gk_reflexes', 'gk_speed', 'gk_positioning' and ‘body_type’* based on dictionary definitions or repeated columns as they add no useful impact in our analysis.

- We converted *overall* ratings into 2 binary classes, with rating > 70 (as many big clubs use this threshold) to recruit their team players and will be treated as our dependent variable.

5. EXPLORATORY DATA ANALYSIS

We first considered various interesting statistics for performing our exploratory data analysis.

- Univariate statistics like Missing values percentage in the whole data to treat the missing values, univariate statistics of the continuous variables (count, mean, std, min, max, skewness, kurtosis, unique, missing, IQR) and their distributions.
- Bivariate statistics: Correlation among the features and T-test for continuous variables and chi-square test & Cramer's V for categorical variables.

5.1. Univariate

We performed univariate analysis on the continuous variables to get the sense of the distribution of different fields in our dataset. According to our observation (mean, std, skewness, kurtosis, etc.), we observed many key features follow a normal distribution. Moreover, the Interquartile range (IQR) was used to detect outliers using Tukey's method.

	count	mean	std	min	max	skewness	kurtosis	unique	missing	IQR	OL
age	17770	25.19392	4.651957	16	45	0.39866	-0.46014	29	0	7	10.5
potential	17770	71.38295	6.145171	48	95	0.236877	0.02395	47	0	8	12
international_reputation	17770	1.117783	0.401085	1	5	3.960282	17.97198	5	0	0	0
weak_foot	17770	2.946427	0.659353	1	5	0.143394	0.620083	5	0	0	0
skill_moves	17770	2.346595	0.749672	1	5	0.172797	-0.0522	5	0	1	1.5
release_clause_eur	16257	4645685	11230302	13000	2.28E+08	7.087941	76.74088	1240	1513	2975000	4462500
nation_jersey_number	1104	12.14312	7.350035	1	87	1.777163	16.37297	27	16666	12	18
pace	15784	67.81532	11.32932	24	96	-0.54839	0.554181	71	1986	14	21
shooting	15784	52.24873	14.03465	15	93	-0.28925	-0.757	78	1986	21	31.5
dribbling	15784	62.26717	10.44266	23	96	-0.59282	0.417391	70	1986	12	18
defending	15784	51.66548	16.34756	15	91	0.33055	-1.09031	77	1986	28	42

Please refer to the attached excel file (eda_n.xlsx) on blackboard.

For the categorical variables, univariate analysis consists of their count, unique values, categories with maximum counts (i.e., top), their frequency and number of missing values they have. From the categorical table, we can see that *player_tags*, *loaned_from*, *nation_position*, *player_traits* have more than 54% of missing values. It would not be easy to impute these with any promising values**.**

	count	unique	top	frequency	missing
nationality	17770	160	England	1656	0
club	17770	679	Arsenal	33	0
preferred_foot	17770	2	Right	13639	0
body_type	17770	10	Normal	10410	0
player_tags	1516	84	#Strength	509	16254
team_position	17547	29	SUB	7593	223
team_jersey_number	17770	2	0	11592	0
loaned_from	1281	342	Atalanta	20	16489
nation_position	1104	28	SUB	576	16666
player_traits	8010	1544	Speed Dribbler (CPU AI Only)	365	9760
rating	17770	2	0	12220	0
attack_rate	17770	3	Medium	12063	0
defence_rate	17770	3	Medium	13216	0

5.2. Bivariate

For continuous variables,

We built a correlation matrix to get a sense of the extent of the linear relationship between rating and other explanatory variables and which variables can be excluded in later stages.[1]

Based on correlation matrix, we observed that some variables are highly correlated with each other.

5.3. T-test

We also performed t-test to check whether the mean of the variables when rating = 1 is significantly different from the mean of the variables when rating = 0. After this stage, we removed some variables which are either not significant or having no correlation at all with dependent variables.

[1] We used the seaborn package in Python to create the above heat map. Please refer appendix 1 for the refernce

5.4. For categorical variables,

We performed a **chi-square test** to check the significance of the variables with dependent variable rating. The table below contains p values corresponding to categorical variables. we obtained that *preferred_foot* is not significant in our analysis.

```
preferred_foot         2.856450e-01
club_new               2.736419e-04
body_type              1.120924e-07
defence_rate           4.002939e-26
attack_rate            6.582139e-54
team_jersey_number     1.948734e-62
nation                 4.307277e-84
Pos                   1.142634e-238
dtype: float64
```

To find the correlation between categorical variables with the dependent variable, we applied **Cramer's V** rule.

V equals the square root of chi-square divided by sample size, n, times m, which is the smaller of (rows - 1) or (columns - 1): **$V = SQRT(X^2/nm)$.**

- ***Interpretation:*** V may be viewed as the association between two variables as a percentage of their maximum possible variation. V^2 is the mean square canonical correlation between the variables. For 2-by-2 tables, V = chi-square-based measure of association.
- ***Symmetricalness:*** V is a symmetrical measure. It does not matter which is the independent variable.
- ***Data level:*** V may be used with nominal data or higher.
- ***Values:*** Ranges from 0 to 1.

In this scenario, we kept columns which showed decent correlation between independent variables and dependent variables. These are 'club_new', 'Pos', 'attack_rate', 'nation'

6. FEATURE ENGINEERING:

- **Re-categorizing/imputing Variables**
  - Since *team_jersey_number, nation_jersey_number* is not actually a continuous variable, we have decided to treat them as categorical variables.
  - We further imputed *team_position* with 'not played' for the missing values and re-categorized the players into *defender, attacker, goalkeeper, resting, Mid-Fielder, substitute, not played,* to reduce 29 unique values to 7 levels.
  - We conjecture that a goalkeeper will have minimum values for *'pace', 'shooting', 'passing', 'dribbling', 'defending', 'physic'* thus imputing with such values.
  - Moreover, 2 variables - *nationality* and *club* have very high cardinality. Based on their volume and event rate, we have re-categorized them into low cardinal variables.

- **Creating Variables:**
  - From the data, we observed that '*player_positions'* gives the idea about players multiple playing positions. so, we have decided to assign individual players with the total count of their availability at different on-field positions into '*playing_positions'.*
  - A player's *work_rate* is given by his attack and defense rate; thus, we have separated them into variables.

- We have also calculated the *term* an individual player will be associated with the club to better understand their loyalty with the club.
- We have also used one-hot encoding to utilize categorical variables in a form that could be provided to ML algorithms to do a better job in predictions.

7. MODEL 1

Here, Y = Rating with population event rate as 31.23 % (which is class 1)

7.1. Logistic Regression:

For the logistic regression model, we first performed the classification without regularization followed by a ridge and lasso regression. L1 regularized logistic regression requires solving a convex optimization problem. However, standard algorithms for solving convex optimization problems do not scale well enough to handle the large datasets encountered in many practical settings.

Objective of Logistic Regression while applying penalty to minimize loss function:

$$Hypothesis: \quad h_\theta(x) = \frac{1}{1 + e^{(-\theta^T x)}}$$

$$L1: \quad J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \quad s.t. \quad \|\theta\|_1 <= C$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{m}\sum_{j=1}^{n} |\theta_j|$$

$$L2: \quad J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \quad s.t. \quad \|\theta\|_2^2 <= C^2$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

$$m = number\ of\ samples, \quad n = number\ of\ features$$

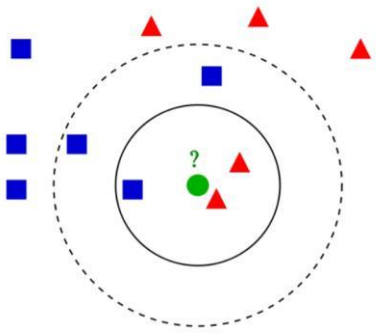
The best result received from running the logistic regression models pre and post regularization (L1 and L2) can be summarized below:

Logistic	Accuracy	AUC	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1 score(0)	F1 score(1)
Train	0.91	0.88	0.92	0.87	0.94	0.83	0.93	0.85
Validation	0.91	0.89	0.93	0.89	0.95	0.83	0.94	0.86
Test	0.91	0.88	0.92	0.87	0.94	0.83	0.93	0.85

7.2. KNN:

kNN is a case-based learning method, which keeps all the training data for classification. One of the evaluation standards for different algorithms is their performance. As kNN is a simple but effective method for classification and it is convincing as one of the most effective methods it motivates us to build a model for kNN to improve its efficiency whilst preserving its classification accuracy as well.

Looking at Figure 1, a training dataset including 11 data points with two classes {square, triangle} is distributed in 2-dimensional data space. If we use Euclidean distance as our similarity measure, many data points with the same class label are close to each other according to distance measure in the local area.



$$h(\boldsymbol{x}') = \arg\max_{y \in Y} \left\{ \sum_{i \in kNN(\boldsymbol{x}')} 1_{[y^{(i)}=y]} \right\}$$

For instance, if we take the region where *k=3* represented with a solid line circle and check the majority voting amongst classes we observe that our data point {circle} will be classified as a triangle. However, if we increase the value of *k* =5 represented by the dotted circle, our data point will be classified as a square. This motivates us to optimize our *k-Nearest Neighbors* algorithms to find the optimal *k* where the classification error is minimal.

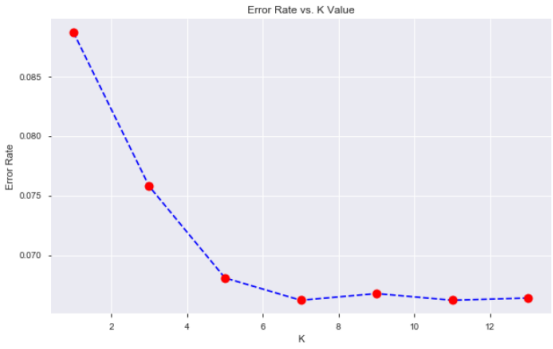
Experiment:

We initially trained our *k*-NN model with *k*=1, with splitting our data into 70% -30% as our training and validation data. From table 2, we observe that training accuracy is 1 which implies that the model fits perfectly, however the accuracy and AUC for the test data is higher than validation data, which is indicative of overfitting therefore we are subjective to perform parameter tuning.

k=1	Accuracy	AUC	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1 score(0)	F1 score(1)
Train	1	1	1	1	1	1	1	1
Validation	0.91	0.884	0.92	0.88	0.95	0.82	0.93	0.85
Test	0.93	0.912	0.94	0.89	0.95	0.88	0.95	0.88

Optimization:

We utilized elbow method to find the least error on training data.



After running for the best *k*, we observed that the least error rate is observed at when *k=7*, Although our optimized results performed better in train and validation, our test AUC has reduced. Even though the accuracy for the test is reduced, we observe that the precision- recall for the same has increased indicating that our model is classifying more class (1) better as that is our target class. (players with greater than 70 rating)

k=7	Accuracy	AUC	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1 score(0)	F1 score(1)
Train	0.95	0.931	0.95	0.93	0.97	0.89	0.96	0.91
Validation	0.93	0.916	0.94	0.91	0.96	0.87	0.95	0.89
Test	0.92	0.894	0.92	0.91	0.96	0.83	0.94	0.87

7.3. DECISION TREE:

The decision tree method is a powerful statistical tool for classification, prediction, interpretation, and data manipulation that has several potential applications in many fields.

Using decision tree models has the following advantages:

- Simplifies complex relationships between input variables and target variables by dividing original input variables into significant subgroups.
- Non-parametric approach without distributional assumptions so, Easy to understand and interpret.

The main disadvantage is that it can be subject to overfitting and underfitting, particularly when using a small data set.

Experiment:

We trained our Decision tree classifier from Sklearn library without passing any parameters. From the table we observed that there is an overfitting of the data thus we must tune our parameters to get optimized results.

	Accuracy	AUC	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1 score(0)	F1 score(1)
Train	1	0.99	1	1	1	1	1	1
Validation	0.92	0.91	0.95	0.87	0.94	0.88	0.95	0.88
Test	0.92	0.9	0.93	0.89	0.95	0.85	0.95	0.88

Optimization:

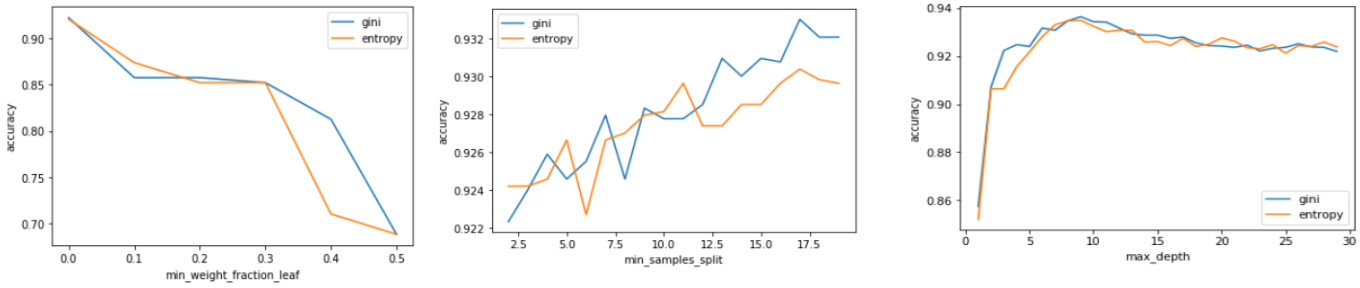
We worked with the following parameters:

- criterion: string, optional (default=" Gini"):

$$Gini: Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

$$Entropy: H(E) = -\sum_{j=1}^{c} p_j \log p_j$$

- max_depth: int or None, optional (default=None):
  The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- min_samples_split: int, float, optional (default=2):
  The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- min_weight_fraction_leaf: float, optional ():
  The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided



From the experiments above, we see that *Gini* is outperforming the *entropy* in all the variants of the experimental parameters. Thus, our criteria are *Gini*. Similarly, we can observe the other parameters for *max_depth = 10, min_samples_split = 17.5, Min_weight_fraction_leaf =0,* Gini gives higher accuracy. Thus, utilizing these parameters, we train our model to observe that there is no overfitting and we can capture more true classes in class 1 category.

	Accuracy	AUC	Precision (0)	Precision (1)	Recall (0)	Recall (1)	F1 score(0)	F1 score(1)
Train	0.96	0.946	0.96	0.94	0.97	0.92	0.97	0.93
Validation	0.94	0.924	0.95	0.9	0.96	0.89	0.95	0.9
Test	0.93	0.919	0.95	0.91	0.96	0.88	0.95	0.89

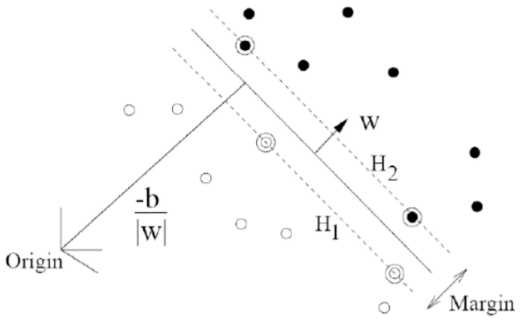
7.4. SUPPORT VECTOR MACHINES:

The folklore view of SVM is that they find an "optimal" hyperplane as the solution to the learning problem. The simplest formulation of SVM is the linear one, where the hyperplane lies on the space of the input data **x**. In this case the hypothesis space is a subset of all hyperplanes of the form:

$f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + b$.

Hard Margin Case:

$$f(x) = \begin{cases} +1, & w * x_i + b \geq +1 \\ -1, & w * x_i + b \leq -1 \end{cases} \quad (1),(2)$$



The maximum margin separating hyperplane objective is to find:

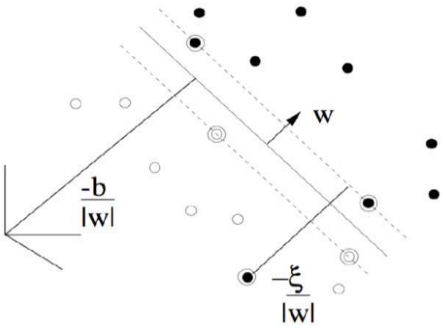
$$\min_{w \in H} \tau(w) = \frac{1}{2} \|w\|^2 \text{ subject to } y_i(w * x_i + b) \geq 1 \ \forall i$$

Soft Margin Case:

Slack variables are part of the objective function too:

$$\min_{w \in H, \xi \in R^m} \tau(w, \xi) \equiv \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i$$

The cost coefficient C>0 is a hyperparameter that specifies the misclassification penalty and is tuned by the user based on the classification task and dataset characteristics.



$$\max_a L_D \equiv \sum_i a_i - \frac{1}{2} \sum_{i,j} a_i a_j y_i y_j x_i * x_j$$

subject to,

$$\forall i \begin{cases} \sum_i a_i y_i = 0 \\ C \leq a_i \leq 0 \end{cases}$$

RBF SVMs

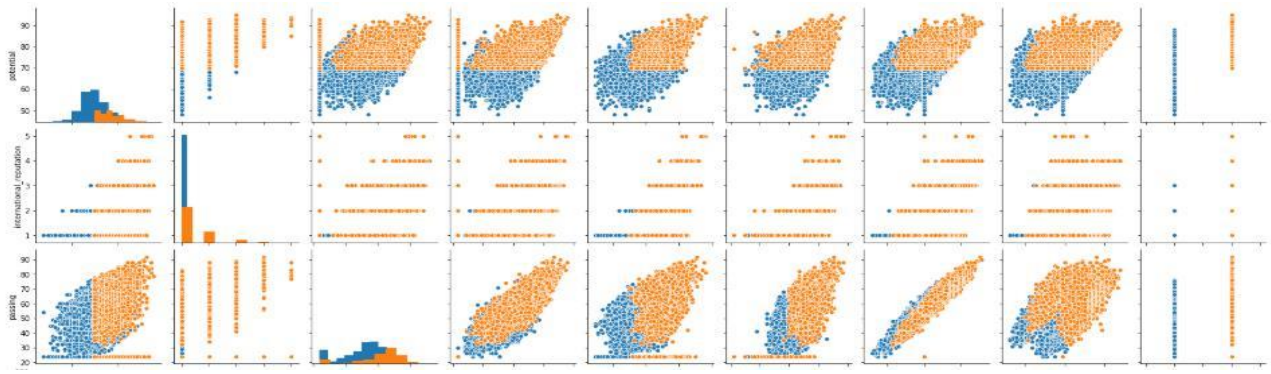
In general, the RBF kernel is a reasonable first choice. This kernel nonlinearly maps samples into a higher dimensional space, so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear. Furthermore, the linear kernel is a special case of RBF since the linear kernel with a penalty parameter $\hat{C}$ has the same performance as the RBF kernel with some parameters (C, γ). The second reason is the number of hyperparameters which influences the complexity of model selection.

Experiments:

We subjected our training data to a linear SVM classifier without training it for soft margins. The results observed does look promising however,

SVM(linear)	Accuracy	AUC	Precision (-1)	Precision (1)	Recall (-1)	Recall (1)	F1 score(-1)	F1 score(1)
Train	0.94	0.92	0.95	0.91	0.96	0.88	0.95	0.9
Validation	0.93	0.914	0.94	0.92	0.96	0.86	0.95	0.89
Test	0.93	0.922	0.96	0.87	0.94	0.9	0.95	0.89

The reason for the good score was that the data was almost **linearly separable** most of the time with very few misclassifications.



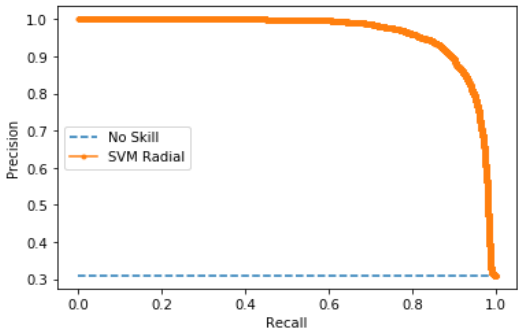
Optimization:
We decided to run a grid search with linear, and radial basis function with varying C and γ to train our model efficiently. From the Grid search we obtained the best estimators as

SVC(C=1, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma='auto_deprecated', kernel='linear', max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001, verbose=False)

And for the radial basis function we got our best estimators as
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf', max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001, verbose=False)

SVM(rbf,C=100,Gamma = 0.001)	Accuracy	AUC	Precision (-1)	Precision (1)	Recall (-1)	Recall (1)	F1 score(-1)	F1 score(1)
Train	0.95	0.94	0.96	0.94	0.97	0.91	0.97	0.92
Validation	0.94	0.93	0.95	0.92	0.96	0.9	0.96	0.91
Test	0.94	0.918	0.94	0.93	0.97	0.87	0.96	0.9

Since the generalization error (expected loss) is used to approximate the population error, we observed the $Error_{val}$ in RBF kernel model is the smallest amongst other models. Also, this is our best model as it fits the data better than the rest of the models.

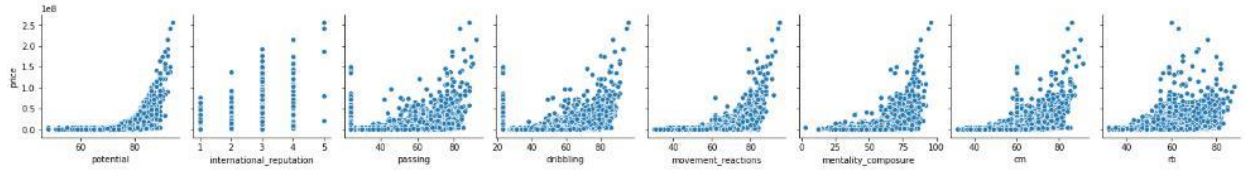


RBF kernel took the data into higher infinite dimensional space which helped our model to stand out. Precision-Recall curve shows us how well the positive class is getting predicted with AUC - 0.961.

8. MODEL 2:

Here, X is same including predicted rating from Model 1 and **Y = Release clause + 52*Wage** as cost to investors. (52 is multiplied as weekly wage is given).

After selecting significant variables, from Univariate and Bi-variate analysis as earlier, we plotted a scatter plot of independent variables with the dependent variables.



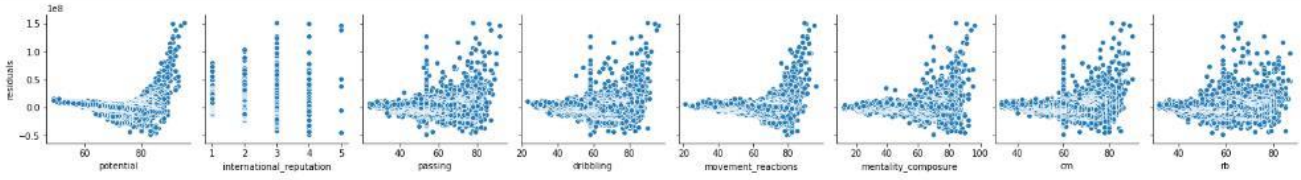
It is clearly visible that they follow a relationship, but it does not seem linear. We confirmed this by developing a linear model.

8.1. Linear Model:

Results:
R square train **0.54**
R square validation **0.55**
R square test **0.54**

R square is the measure of closeness to perfect prediction. Here, R square is not good.

Checking Linearity from residuals: Data should be randomly scattered. But here, we figured out that they are not random. This means a linear model would never be a good choice to fit this model.

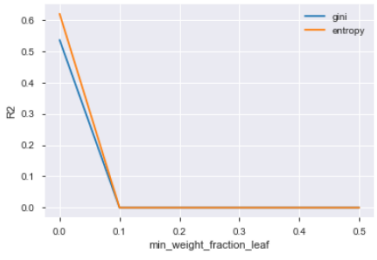


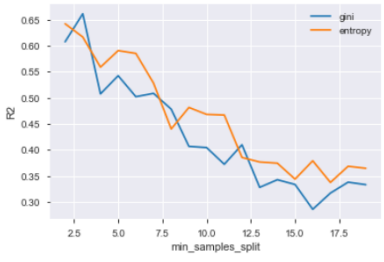
8.2. Decision Trees: This was the better choice than linear models in this scenario.

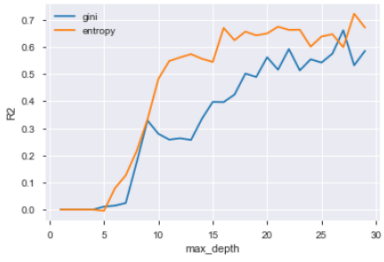
Results (Baseline):

Train Data: R square - 0.99 and RMSE - 0.05
Validation Data: R square - 0.54 and RMSE - 8.05
Test Data: R square - 0.59 and RMSE - 7.35

There was a clear indication of over-fitting. Model was not performing as expected. Therefore, we tried grid search based on min_split, tree_depth and min_weight_fraction_leaf and learning criteria.







As shown above, Entropy performed better with min_split=3 and max_depth=15.

Results after Grid Search: (Main Model)
Train Data: R square - 0.85 and RMSE - 4.40
Validation Data: R square - 0.69 and RMSE - 6.59
Test Data: R square - 0.70 and RMSE - 6.26

R-squared value seems far better now. RMSE value is also low and the problem of overfitting is also solved. **Hence, Decision Trees performed better here in order to predict the cost to investors.**

Final Strategy:

Final step was to make a strategy to pick players for our team keeping in mind:

- Rating should be greater than 70 (means class 1)
- Budget - 1 billion Euros and number of players around 30.

Firstly, we selected only the players who had ratings greater than threshold of 70. Number of players left - **5276**

Secondly, we performed some analysis like the decile analysis of the cost to investors. We made some buckets each having approximately 30 players from the remaining pool and sorted those buckets based on cost to investors in descending order.

S.No.	Cost Bucket	Minimum	Median	Maximum	Total Cost (in Euros)
1	(85800000.0, 165380000.0]	93396000	109600000	165380000	3457716000
2	(73940000.0, 85800000.0]	74400000	85560000	85800000	2031148000
3	(65860000.0, 73940000.0]	66764000	72640000	73940000	2777488000
4	(60706352.941, 65860000.0]	60720000	63960000	65860000	2052224000
5	(51112000.0, 60706352.941]	53052000	56360000	60604000	1625460000
6	(45537647.059, 51112000.0]	45640000	47364000	51112000	1588064000
7	(40205411.765, 45537647.059]	40240000	43260000	45060000	1319120000
8	(36876000.0, 40205411.765]	36992000	38924000	40072000	1164392000
9	(35640000.0, 36876000.0]	35888000	36386000	36876000	1089476000
10	(34367058.824, 35640000.0]	34380000	35156000	35640000	1158128000
11	(32088000.0, 34367058.824]	32624000	33840000	34336000	945816000
12	(30356000.0, 32088000.0]	30536000	31076000	32088000	939504000
13	(28676000.0, 30356000.0]	28836000	29656000	30356000	946108000
14	(27872000.0, 28676000.0]	27912000	28548000	28676000	936908000
15	(26936000.0, 27872000.0]	27352000	27476000	27872000	771980000
16	(26208000.0, 26936000.0]	26248000	26756000	26936000	879388000
17	(25096000.0, 26208000.0]	25136000	25248000	26208000	815168000
18	(23924000.0, 25096000.0]	23936000	24152000	25096000	753472000
19	(23028000.0, 23924000.0]	23048000	23558000	23924000	704332000
20	(22200000.0, 23028000.0]	22288000	22686000	23028000	678992000

Here, we can observe that the amount needed to pick the whole team from the first bucket is **3.45 billion Euros** (which is out of budget). That means we can't pick top 30 players directly and the amount needed to pick the team from the 11th bucket is **0.945 billion Euros** (which is in our budget). However, it would be a wrong strategy to pick all the players from this bucket only as we'd leave almost 300 high valued players who are above this bucket. So, the best solution is to pick 8-10 core players from the top buckets and rest of the players from medium and low-valued buckets.

This decision can be easily made by the above analysis and it is up to the investors and team managers to decide that what kind of players they want in their team.

9. CONCLUSION:

In this work, we constructed 2 models that utilizes Machine learning algorithms to benefit investors while simultaneously capturing the meaningfully classifying players as good performers and then regressing them in the budget of the investor. The result, classification and regression fitting, is a new selection model for the supervised learning of players that outperforms other teams. Ultimately, we have narrowed down the selection process of a player within a club, which is rather better than selecting at random.

Future Scope: We can also implement Time Series techniques. As our dependent variables – rating and cost both depends on previous years data. For example – If a certain player has a rating of 85 in Dec'19, his rating in Jan'20 would be around 85 +/- 3. Therefore, Time Series techniques might be useful for this data.

References:

1. Guo, Gongde & Wang, Hui & Bell, David & Bi, Yaxin. (2004). KNN Model-Based Approach in Classification.
2. Yan-yan SONG, Ying LU. (2015). Decision tree methods: applications for classification and prediction.
3. https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680
4. Apostolidis-Afentoulis, Vasileios. (2015). SVM Classification with Linear and RBF kernels. 10.13140/RG.2.1.3351.4083.

APPENDIX 1: