ECUBE ANALYTICS

POLYNOMIAL REGRESSION AND REGRESSION SPLINES

In this article series with accompanying code, there are two parts:

1. **POLYNOMIAL REGRESSION**
2. **REGRESSION SPLINES**

This is the first part. Articles are meant to give a feeling as if the reader is working in Industry, gets a new dataset, struggles with it using different approaches, learns in the process why a particular approach does not work, looks for better alternatives and finally finds the best approach. **Currently I do not discuss deployment but in future I intend to explain that too for this article series.**

A. Here, we get a 'tricky' dataset and do EDA and plot probability distribution plots for response variable to check if normality assumption of linear regression is valid. Here it is not valid.

B. Next we apply transformations on response-variable to see if that can bring 'normality'. That too fails.

C. Then we do outlier and NaN detection and treatment.

D. For NaN imputation 3 different methods are tried:

1. NaN imputation by Pandas. (code link-1)

ECUBE ANALYTICS

2. NaN imputation by SKlearn (Simple, iterative, KNN) (code link-2)
3. **NaN imputation by a latest technique (2020-2021): LGBM imputation** (code link-3)

E. For outlier removed and NaN imputed data, we do Bi variate analysis (code link-3), including plotting the correlation-plot, **for linear correlation**, which confirms poor linearity between predictors and response **and** high multicollinearity among predictors.

F. We then introduce a new **non-linear correlation coefficient** called **Chatterjee Correlation Coefficient (CCC)**. Even these non-linear correlation coefficients between predictors and response variable are not that great, thus mentally preparing us to be ready to go beyond even polynomial regression if needed.

G. This (and many data-sets in real life) contains some categorical variables and usual tendency is to encode them directly. But the correct approach is to first check if they affect the outcome variable. So, the **article takes a detour** and discusses Anova Test to decide if categorical variables are significantly affecting the response variable. Here we find that they do. So, we

do one hot encoding for them.

H. Then we do (code link-3) a train-test split for LGBM imputed dataset and obtain an OLS model. Again there is a lengthy detour to discuss the terms in its output to make the reader realize how rich a source of information this output is.

I. Finally, as the RMSE and R-squared for OLS are average, we do polynomial regression. Its plots help us to skip some variables from the list of predictors. Then we make polynomial models and see that RMSE and R-squared improves but not that great.

J. Still, few questions remain unanswered which will be answered in the next article (regression splines) in this series.

I. POLYNOMIAL REGRESSION

Here we take the example of the application of multiple linear regression in sports analytics: Let's say there is a game of basketball ahead. From practice sessions we have got continuous variables $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$,....., $\mathbf{X_n}$. Similarly **Y** is the number of points which the team will

score that we want to predict which is also a continuous variable. Thus this is a case of regression.

Business Problem:

Make a regression model using quantitative independent variables $\mathbf{X_1, X_2, X_3,.....Xn}$ that can predict the points scored **Y** in a match.

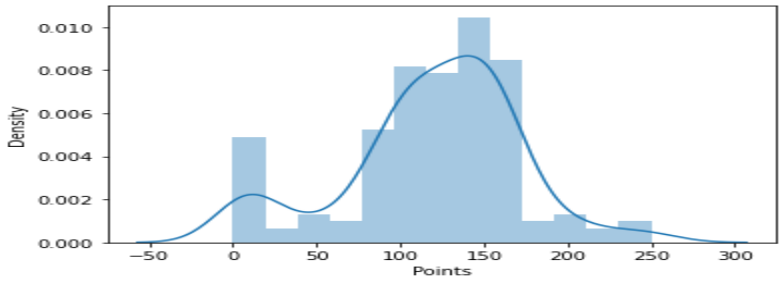
Data Link:

Jupyter Notebook links: link-1, link-2, link-3

UNIVARIATE ANALYSIS:

A. Response Variable Transformation And Analysis:

First let us analyze the response variable 'Points' .We plot the **distplot** (histogram plus probability distribution plot) using seaborn library.

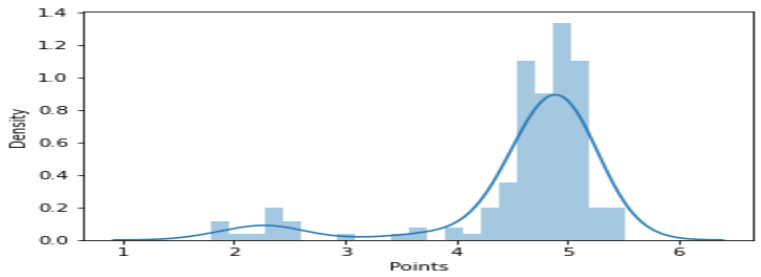


The above plot shows that distribution is **NOT** perfectly normal. Further, there are some observations in which the target variable 'Points' is 0. Hence the distribution even goes towards negative values of 'Points'. (It does **not**

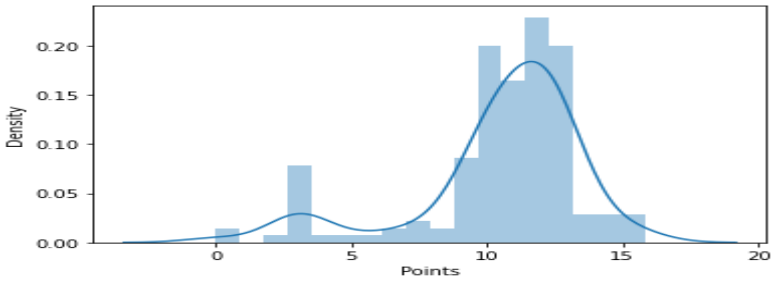
mean **there are** actually negative values in the "Points" column). Output of **df1.describe()** above also showed that the minimum value of points is zero. Further the peak value of "Points" is between 125-150.

Let's apply some transformations **on response variable** 'Points' and plot the distplot again:

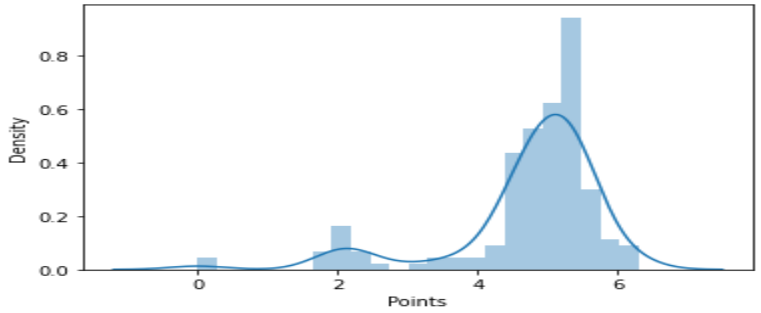
- **Logarithmic Transformation:**



- **Square Root Transformation:**

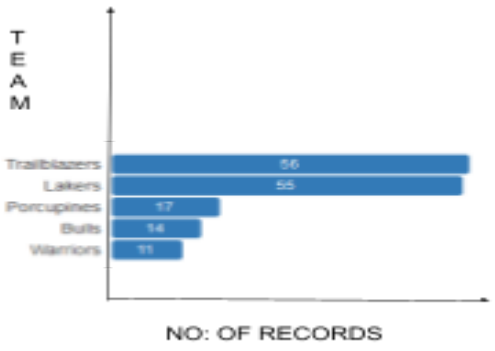


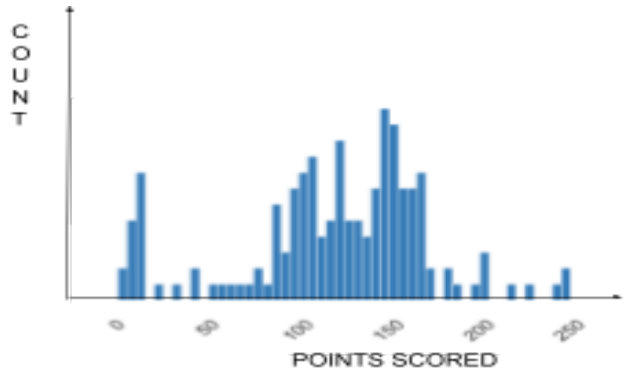
- **Cube root Transformation:**



Thus by all three transformations tried above, **non-normality can not be removed.**

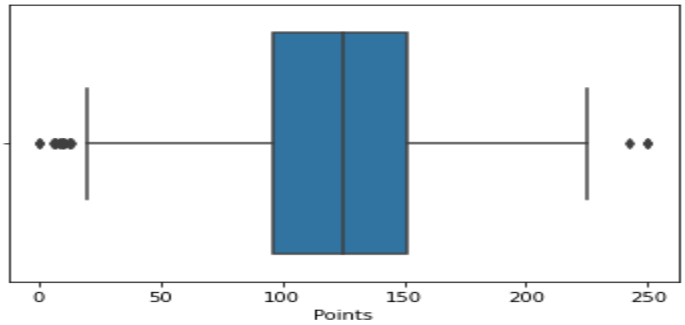
The 'Team' is a categorical variable with six categories. Its horizontal bar plot is shown below. 'Points Scored' or 'Points' is the response variable. Its histogram is also shown below.





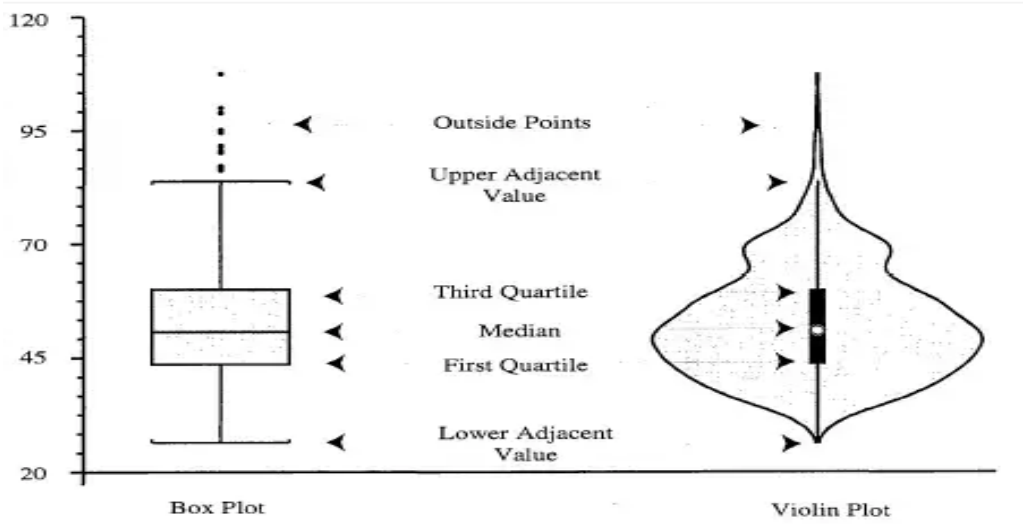
B. Outlier Detection And Removal , Variance Analysis

The boxplot of 'Points' shows outliers on both sides. Sometimes outliers also have meaning. Thus, we leave them for now.



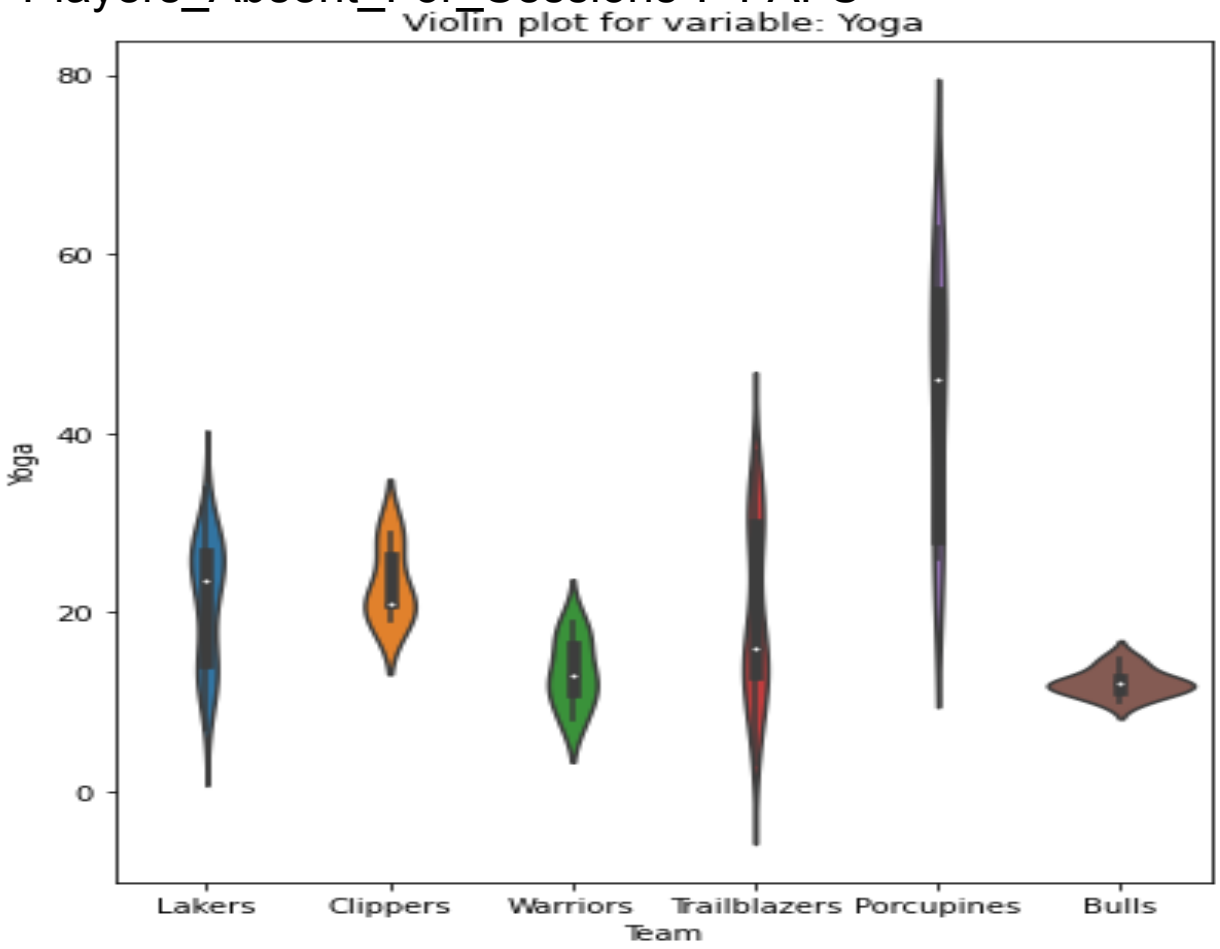
ECUBE ANALYTICS

Similarly we draw box plots and violin-plot for anomaly detection (mainly outliers) in predictor variables. **The boxplots provide the analysis across the entire dataset but violin-plot will provide analysis for each team separately.** The violin-plot (boxplot inside kernel density plots) has some clear advantages over box plot. So, below only they are discussed. The jupyter notebook, however shows both types of plots.The figure below shows **important features of both types of plot**:



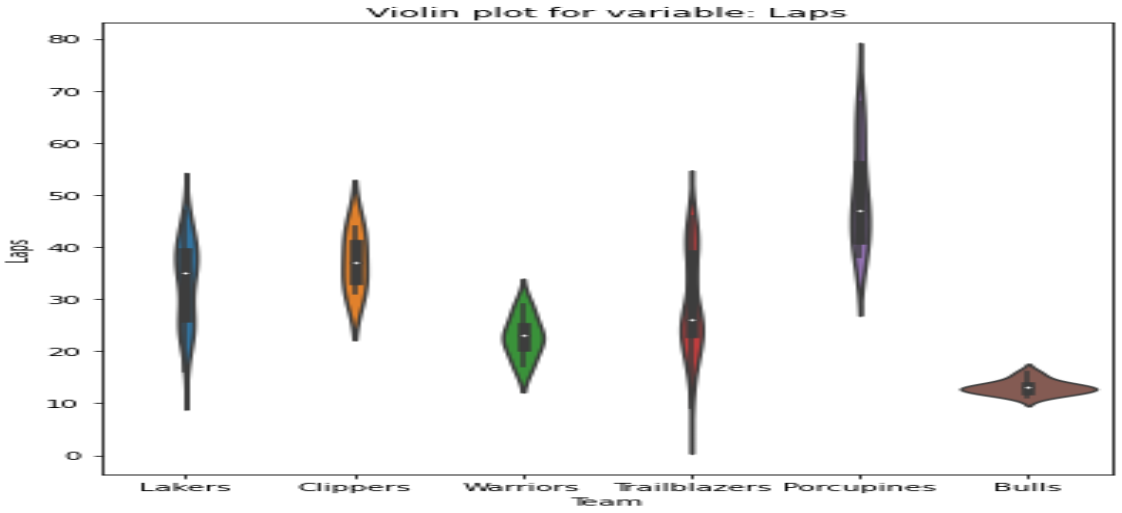
For more on violin plot readers are strongly recommended to refer to the following links. Link-1, Link-2, Link-3. The violin-plots we obtained for all predictors are shown below:
NOTE: Independent variable-names are shortened:
"Weightlifting_Sessions_Average": "WL" (**not shown below**)

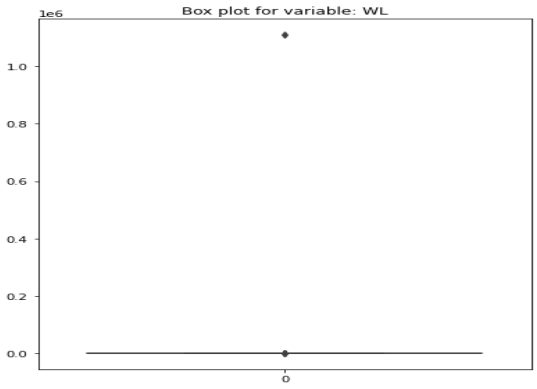
"Yoga_Sessions_Average": "Yoga",
"Laps_Run_Per_Practice_Average": "Laps",
"Water_Intake": "WI",
"Players_Absent_For_Sessions": "PAFS"

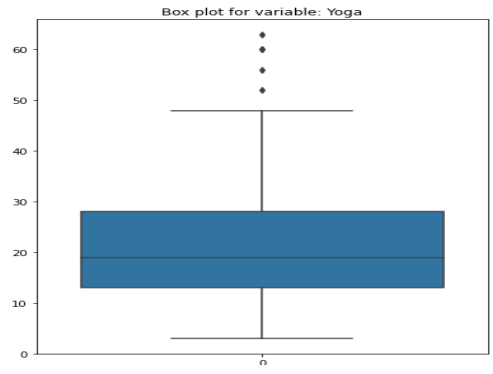


In the above violin-plot the ‘bulges’ signify the probability distribution and the white dot shows the median for the respective team. A violin-plot like that for ‘Bulls’ shown

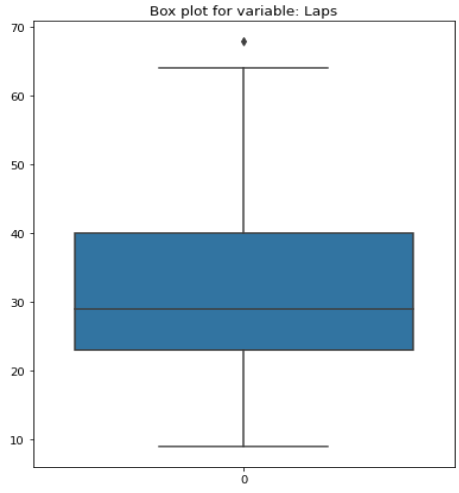
above is a ‘good’ violin-plot in which there is nearly normal distribution and probability of most data is around median. Its width is less means data has less variance around the median. The plots for Clippers and Warriors also show less variance but distribution has multiple peaks. The plots for ‘Lakers’, ‘Trailblazers’ and ‘Porcupines’ are showing very high variance and also non-normality. The violin-plot for the ‘Laps’ variable also is ‘good’ for the ‘Bulls’ team.

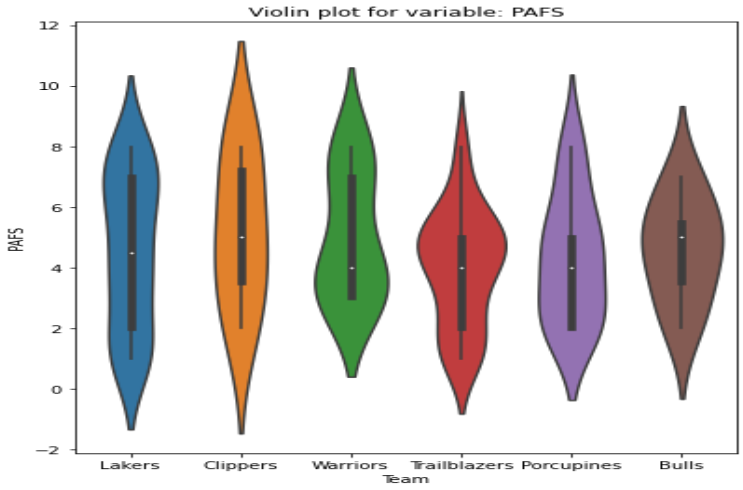




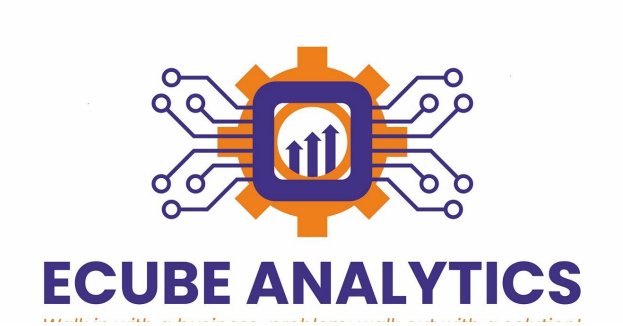


Other violin-plot are not shown here but [jupyter notebook] has all. Now let's discuss box plots for predictor variables. The above boxplot for Weightlifting_Sessions_Average(WL), has one huge outlier. Similarly variable 'Yoga' has 4 outliers and if we see a violin-plot for 'Yoga' on p-9, the team 'Porcupine' seems responsible. **Hence boxplot and violin-plot are often seen side by side as in jupyter notebook**.





The box plot for 'Laps' has one outlier as shown above and again the team 'Porcupine' seems responsible from violin-plot on p-10. The boxplots for **WaterIntake**(**WI**) and **PlayersAbsentForSessions**(**PAFS**) show no outliers (**see jupyter notebook**). Also, the violin-plot for '**PAFS**' shows



that variance for each team is in range around 2-8 with median value equal to 4. So, this variable is behaving 'well' across teams so far as variance is concerned. However, normality wise only for the team 'bulls' the data seems to be behaving well. We also obtained the index and the value of the outliers for each column using a single function. The results are as below:

The index and the value of outliers in column WL are:

8 1111111.0
142 56.0
143 56.0
144 59.0

The index and the value of outliers in column Yoga are:

140 52.0
141 56.0
142 60.0
143 60.0
144 63.0

The index and the value of outliers in column Laps are:

144 68.0

The index and the value of outliers in column WI are:

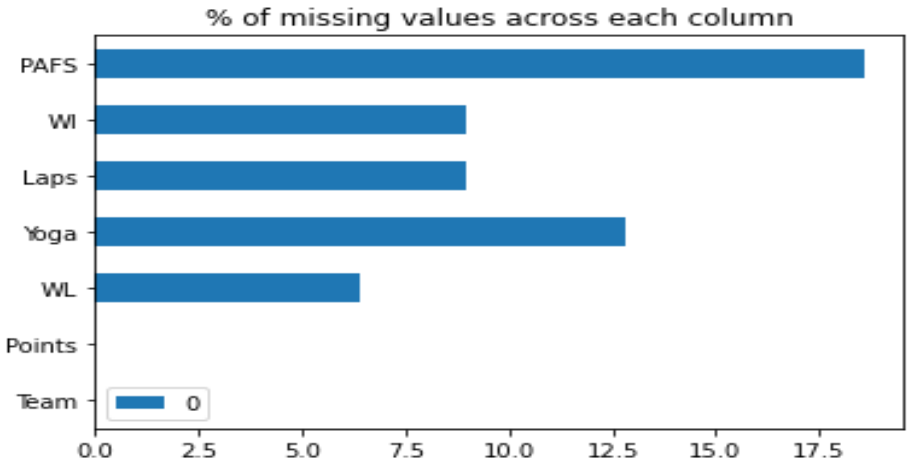
The index and the value of outliers in column PAFS are:

We can drop rows 142,143,144 having outliers in more than one predictor, while for other rows we can use replacement of current value by median/mean/Nan etc.

ECUBE ANALYTICS

C. Nan Value Imputation

After trying our hand at transformation of response variables (where we failed to achieve anything significant) and detecting and removing outliers (which we successfully did), we focus on finding the percentage of missing values. We get results as plotted in graph below:



Next is the issue of replacing these NaN values. For that one can use **sklearn's imputer** (simple or iterative or KNN), or **Pandas methods Ffill and Bfill with groupby** or use of **Light Gradient Boosting Model** (**LGBM**).

C-1 NaN imputation By Pandas (code link-1)

In the first Jupyter Notebook, we first do outlier treatment and then replace NaN using PANDAS by the mean value of each column. However, this dataset contains records for

many teams ['Lakers' ,'Clippers', 'Warriors', 'Trailblazers', 'Porcupines' and 'Bulls']. So **NaN values in records for a given team must be replaced by the mean value of that column for that team and not by the overall mean of the whole data**. Hence we first found column wise means for each team separately in a dictionary. Then we divided the dataframe into many horizontal subsets, each containing records for a single team only. Finally we defined a function which replaces NaN values in a given subset data frame, in a given column , for a given team by the mean value of that column for that team. On concatenating back small dataframes , without NaN values, row-wise, we got a master data frame i.e. df_master. Once imputation is done we can continue univariate analysis. Most important is probability distribution analysis of predictor variables for checking **'normality'** assumption. As the code and the distribution plots are already present in the jupyter notebook **(code link-1)**, we just note down the results in the table below.

PREDICTOR	TRANSFORMATIONS TRIED	BEST LOOKING TRANSFORMATION
WL	Log, square root, cube root	Square root

YOGA	Log, square root, cube root	Square root
LAPS	Log, square root, cube root	Untransformed
WI	Log, square root, cube root	Square root
PAFS	Log, square root, cube root	Untransformed
It is important to note that we have just tried various transformations but not applied and kept df_master as it is so far.		

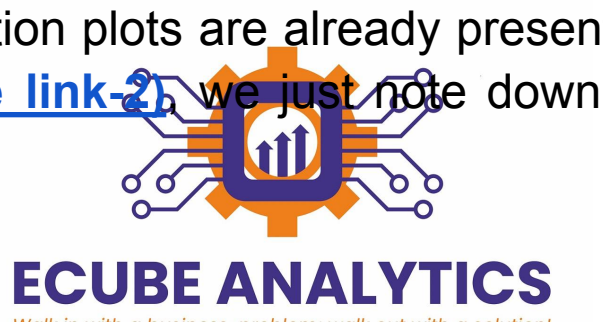
C2-NaN imputation By SKLEARN (Simple, Iterative,KNN)

In the second Jupyter Notebook (code link-2), we first replace NaN using **SKLEARN SIMPLE IMPUTER.** This is much easier compared to PANDAS IMPUTER. We then check for normality. Again, the code and the distribution plots are already present in the jupyter notebook, we just note down the results in the table below.

PREDICTOR	TRANSFORMATIONS TRIED	BEST LOOKING TRANSFORMATION
WL	Log, square root, cube root	Square root Or Untransformed
YOGA	Log, square root, cube root	Untransformed
LAPS	Log, square root, cube root	Square root Or Untransformed
WI	Log, square root, cube root	Square root Or Untransformed
PAFS	Log, square root, cube root	Untransformed
It is important to note that we have just tried various transformations but not applied and kept df_master as it is so far.		

Then we did an imputation by **SKLEARN ITERATIVE IMPUTER** and again did normality check. Again, the code

and the distribution plots are already present in the jupyter notebook (code link-2), we just note down the results in the table below.



PREDICTOR	TRANSFORMATIONS TRIED	BEST LOOKING TRANSFORMATION
WL	Log, square root, cube root	Square root Or Untransformed
YOGA	Log, square root, cube root	Untransformed
LAPS	Log, square root, cube root	Square root Or Untransformed
WI	Log, square root, cube root	Square root Or Untransformed
PAFS	Log, square root, cube root	Untransformed
It is important to note that we have just tried various transformations but not applied and kept df_master as it is so far.		

Finally, we did an imputation by **SKLEARN KNN IMPUTER** (code link-2) and again did a normality check. Again, the code and the distribution plots are already present in the jupyter notebook, we just note down the results in the table below.

PREDICTOR	TRANSFORMATIONS TRIED	BEST LOOKING TRANSFORMATION
WL	Log, square root, cube root	Square root
YOGA	Log, square root, cube root	Square root

LAPS	Log, square root, cube root	Untransformed
WI	Log, square root, cube root	Square root
PAFS	Log, square root, cube root	Untransformed
It is important to note that we have just tried various transformations but not applied and kept df_master as it is so far.		

C-3 NaN imputation By LGBM IMPUTER

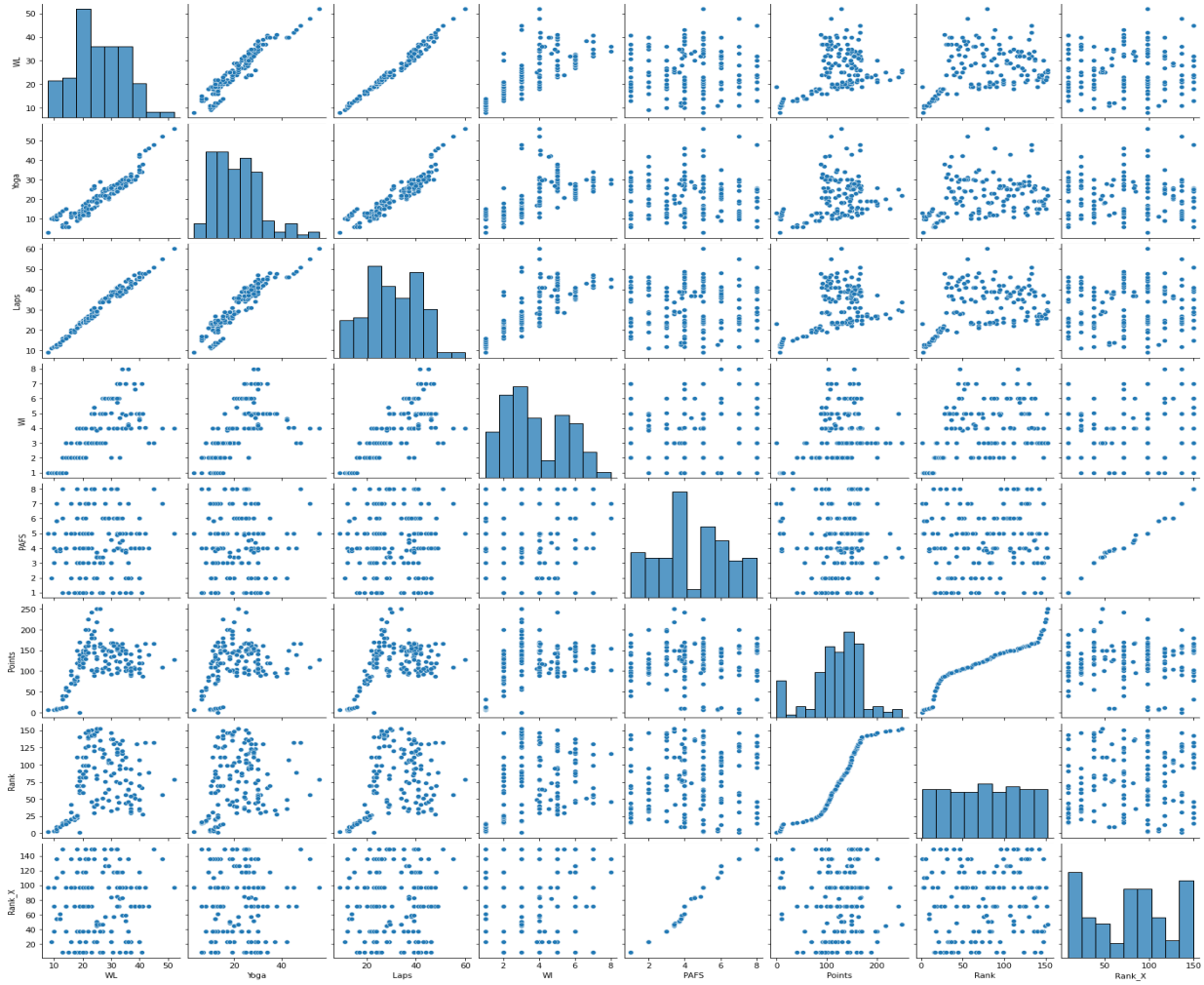
At last we did NaN imputation by **LGBM IMPUTER**. Just like Sklearn's iterative imputer internally uses a regression model, this method uses a light gradient boosting model internally. credit:https://github.com/analokmaus/kuma_util. After imputation, we plot distplots for each predictor variable. Again, the code and the distribution plots are present in the jupyter notebook (**code link-3**), we just note down the results in the table below.

PREDICTOR	TRANSFORMATIONS TRIED	BEST LOOKING TRANSFORMATION
WL	Log, square root, cube root	Square root
YOGA	Log, square root, cube root	Square root Untransformed
LAPS	Log, square root, cube root	Untransformed Square Root
WI	Log, square root, cube root	Square root
PAFS	Log, square root, cube root	Untransformed square root
It is important to note that we have just tried various transformations but not applied and kept the final data frame as it is so far.		

ECUBE ANALYTICS

BI-VARIATE ANALYSIS

Bi-variate analysis must be done after removing outliers and then imputing NaNs. **We finally preferred the LGBM imputed data frame** as in the jupyter notebook. We first applied Panda's automated profiler on a cleaned dataframe, without outliers and missing values. The scatter-plots obtained are as below.



Weak positive correlation is seen in **‘Points’** Vs ‘**WL**’, ’**Yoga’**, and **'Laps'**. for smaller values of Predictors. But for higher values of predictors , variance increases . So, the basic condition of MLR i.e. homoscedasticity is violated. Further linearity is also clearly violated in these three plots. In the last two plots also linearity is not seen. Correlation-plot for final dataframe after outlier removal and and NaN imputation is shown below:

Plot shows multiple problems:

A. Very high degree of multicollinearity: For example,

- WL has 94% correlation with Yoga and 99% correlation with Laps.
- Yoga has 93% correlation with Laps
- Laps has 78% correlation with WI
- WI has 72% correlation with WL

B. The linear correlation for response variable 'Points' with 'WI' is 0.43, that of 'Laps' is 0.48 and thaThet of 'WL' is 0.45. Other values are even smaller.**Thus there is not much linear relation between response variable and predictors**.

C. With the above correlation plot we must note that it may be slightly misleading as it contains data for all the teams in a consolidated manner.

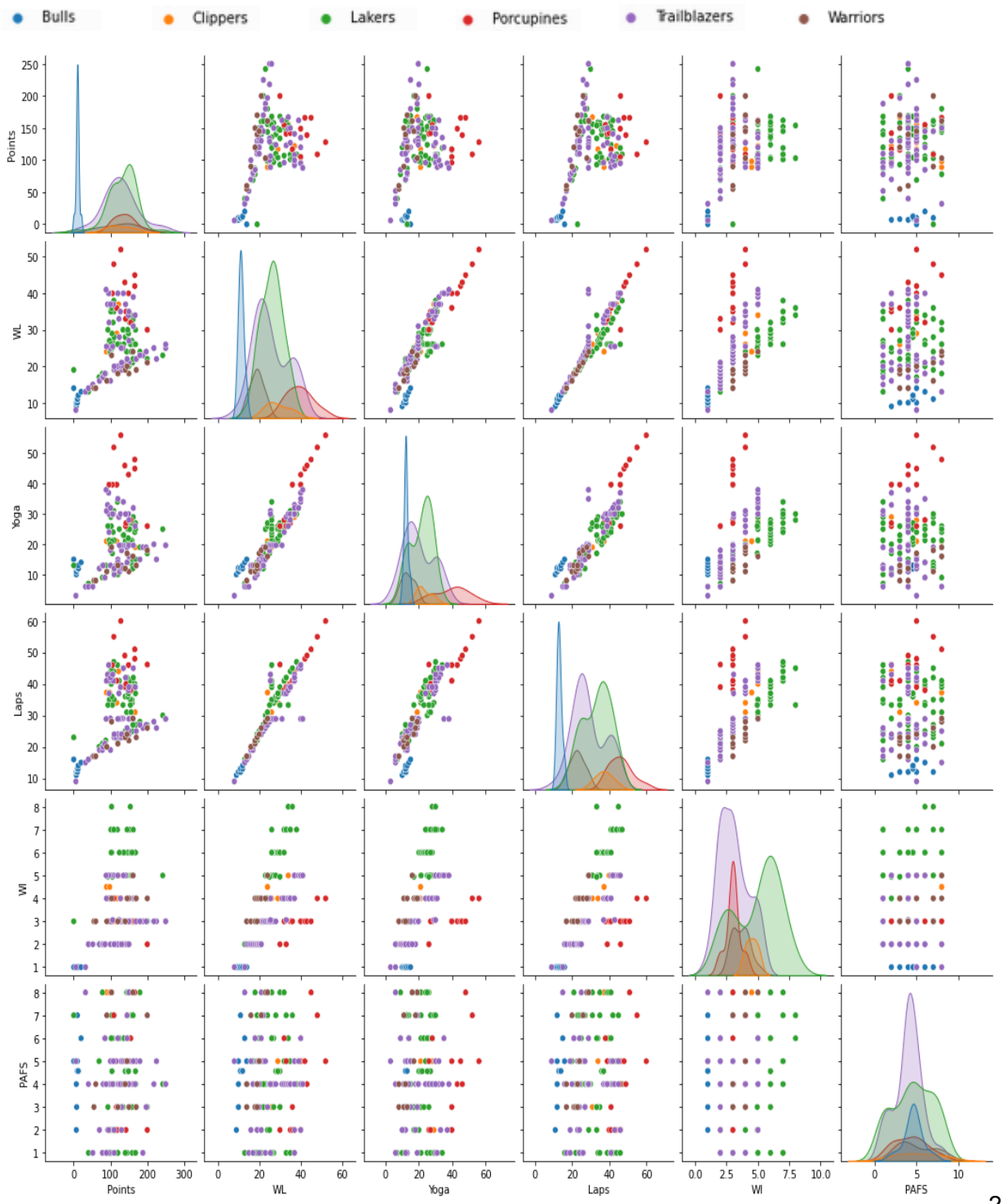
Ideally we must divide the dataframe into smaller dataframes with records for each of the six teams, and plot correlation matrix and scatter plot for each team separately. In jupyter notebook **(code link-2)** , we even did this but found that the data continues to 'misbehave' even at individual team level in three respects:

A. The linear correlation between response variable ‘Points’ and ‘Predictors’ is never above 30% for any team except ‘Warriors’. Even for ‘Warriors’ the highest correlation is 69%. In short, **linearity assumption is not valid either at consolidated level or at individual team level**.
B. Correlation amongst predictors is very high in several cases for each team. In short, **multicollinearity is present both at consolidated level and at individual team level.**
C. In Scatter plots of response variable Vs predictors **linearity and homoscedasticity are violated for most scatter-plots**.

We do not reproduce the plots for each team here as they are already present in the jupyter notebook. However, one pair plot is shown below. **One needs to see dots of one color at a time to study one team**.

(P.T.O.)

ECUBE ANALYTICS



After doing this much efforts on outlier removed and NaN imputed dataset i.e., trying variable transformation and scatter point and distplot analysis for the entire data-set as a whole and at individual team level, we understood our data much better and concluded that there is no chance of linear relationship between response variable and predictors.

So, now we focussed on nonlinear coefficients. Here we introduced a new correlation coefficient **ξ** (Xi=pronounced 'zee') proposed by the 2019 paper of Prof. Sourav Chatterjee of Stanford university. **ξ** should be understood as a measure of independence rather than measure of correlation. Value of **ξ** ranges from **ξ=0** (meaning two variables are totally independent) to **ξ=1** (maximum correlation- **that may be linear or nonlinear**- between two variables). Here, if N are the no: of observations in the dataset then as N varies from 0 to ∞**, ξ** varies from 0 to 1. So, unlike Pearson's or other linear correlation coefficients which vary from -1 to 1, **ξ** never attains negative value. Hence, **ξ** can not decide if two variables are positively or negatively correlated. **It just decides how much independent they are**.

ECUBE ANALYTICS

A Brief Discussion Of Chatterjee Correlation Coefficient (CCC) Formula

Let (X,Y) be a pair of random variables, where Y is not a constant.Let $(X_1,Y_1),(X_2,Y_2),.....,(X_n,Y_n)$ **[n≥2]** be i.i.d. pairs with the same law as (X,Y). First rearrange the data by ranking X such that $X_{(1)}\leq\ldots\leq X_{(n)}$. Let **ranks Y_i [number of j such that $Y_{(j)}\leq Y_{(i)}$]** be r_i and that of Y_{i+1} is r_{i+1}.

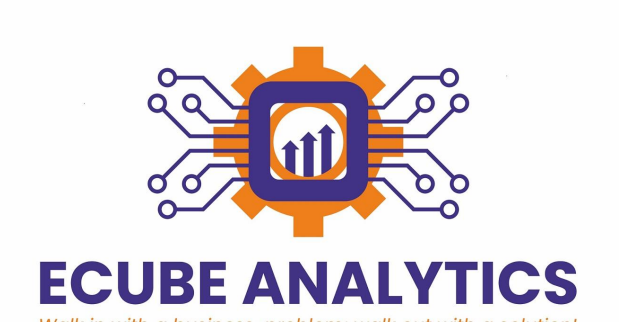
A. If there is no tie in the above ranking,

$$CCC = \xi_n(X,Y) = 1 - \frac{3\sum_{i=1}^{n-1}\left|r_{i+1} - r_i\right|}{n^2-1}$$

B. If there is a tie in the above ranking, first ties are broken **uniformly at random** and apart from r_i we also define l_i **[number of j such that $Y_{(j)}\geq Y_{(i)}$]**

$$CCC = \xi_n(X,Y) = 1 - \frac{n\sum_{i=1}^{n-1}\left|r_{i+1} - r_i\right|}{2\sum_{i=1}^{n} l_i(n-l_i)}$$

Fortunately we do not need to calculate the CCC or **ξ** manually. There is a ready package in both python and R



(XICOR) which we can use. There is also an excellent tool to visualize CCC or **ξ** here. First take noise=0, base function as **sine** and no of observations N=300 and check that **ξ = 0.969** indicating there is **very high nonlinear correlation** between 'Y' and 'X' variable. Now make noise=0.05 and check that **ξ = 0.723.** Repeat the same for base function 'linear' and verify that CCC or **ξ works good for 100% linear.** Finally, for the circle (i.e. 100% non linear) as base function verify that we can not get **ξ** more than 0.22 even with N=2000 observations and various values of noise. This is the reason, in such cases we use algorithms like SVM, neural networks etc.

One can read about various correlation coefficients here. Particularly important is discussion about many pros and a few cons of using CCC instead of conventional coefficients.

The values of CCC we found using XICOR package are:

CCC for WL and Points is:0.407
CCC for Yoga and Points is:0.276
CCC for Laps and Points is:0.364
CCC for WI and Points is:0.230
CCC for PAFS and Points is:0.108

ECUBE ANALYTICS

In addition to ξ/CCC, we also need to understand the one way Anova test before we go to polynomial regression.

ONE WAY ANOVA (Analysis Of Variance) (code link-3)

Dataset has a categorical variable 'Teams' with 6 groups. Instead of directly **ASSUMING** that categorical variable **HAS** effect on response variable, and doing its encoding, we should first do ANOVA analysis. ANOVA is used to check **whether** in two different groups (Say [bulls, Lakers) of a given categorical variable (say Team), **changing Team from bulls to Lakers has an impact on the 'Points Scored'. i.e.** whether the 'Team' affects the 'Points' and should be considered a predictor or not.

Following Terminology Is Required For Anova Test

(A) Difference Among Groups -> **Treatment**

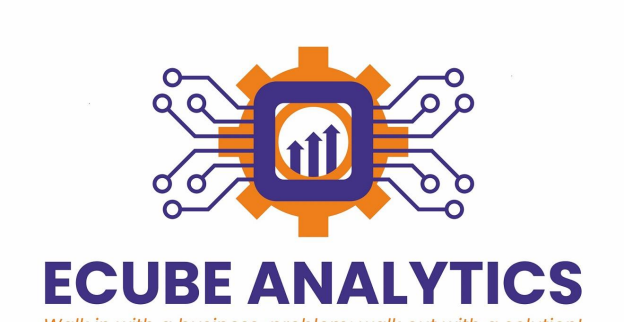
(B) Difference Within Groups -> **Error**

(C) **TOTAL ERROR = TREATMENT + ERROR**

SSTO = SSTR + SSE

DOF **(n-1)** = DOF**(g-1)** + DOF **(n-g)** **(g=no of groups)**

Mean squared value of **SSTO,SSTR OR SSE =** $\frac{Error^2}{DOF}$



Assumptions For Anova

(1) RANDOMNESS: The samples from each group are collected randomly

(2) INDEPENDENCE: The samples are independent (not correlated)

(3)NORMALITY: samples should have normally distributed data-points

(4) HOMOGENEITY: Homogeneity of variance i.e. ratio $\frac{Largest\ Std\ Dev}{Smallest\ Std\ Dev} \leq$ **some domain dependent value V.** In other words: largest Std Dev should be at the most equal to V times smallest Std Dev or less. For sports domain, where many unpredictable events occur V=8-10

Anova Test is done in following steps:

(1) Take independent samples from each group.

(2) Compute means of response variable for every Team.

(3) Compare mean within groups and amongst groups.

(4) Decide if means are equal (i.e. 'Team' has no impact on 'Points') or unequal (i.e. 'Team' has an impact on Points) based on **p and alpha** or **F and F-critical.**

(P.T.O.)

ECUBE ANALYTICS

HOW TO DO ANOVA F-TEST

If group means are $\mu_1, \mu_2, \ldots\ldots\ldots, \mu_k$ then

$H_0: \mu_1 = \mu_2 = \ldots\ldots\ldots = \mu_k$ **null hypothesis**

$H_1: \mu_1 \neq \mu_2 \neq \ldots\ldots\ldots \neq \mu_k$ **alternative hypothesis**

We find $F^* = \frac{MSTR}{MSE}$ and then find $F_{critical}$ **either from the F-Distribution** table OR **from output of Anova test.**

1. $F^* \leq F_{critical}$ $(or\ p > \frac{\alpha}{2})$ then accept null hypothesis.

   So, means **are equal**, or changing the group of categorical variable **has no impact** on response variable.

2. $F^* > F_{critical}$ $(or\ p <= \frac{\alpha}{2})$ then we reject the null hypothesis. So, **means are unequal**,or changing the group of categorical variable **has an impact** on response variables. **We must include the categorical variable in our model**.**Usually $\propto$= 0.05 for one tailed test and 0.025 for two tailed test**

ECUBE ANALYTICS

STANDARD OUTPUT OF ANOVA TEST

The standard output of Anova Test from various software packages is a table as below. Here the values shown in the table in blue color/italics are with the example of our basketball dataset with an LGBM imputer. (code_Link)

	SS	DOF	MS	F	P Value	F critical
Between Groups	***77356.6***	***5***	***15471.33***	***10.3615***	***0.0***	***2.775402***
Among Groups	***94068.8***	***63***	***1493.16***			
Total	***171425.5***	***68***	***2520.96***			

Since F > F critical or p<∝/2(0.05), reject the null hypothesis. So, **means are unequal**,or changing the group of categorical variable **has an impact** on response variables. **We must include the categorical variable 'Team' in our model**. Hence we used one hot encoding to convert 'Team' to a numeric variable.

Then we divided the data-set into train and test, did LGBM imputation in the train dataset and used statsmodel.OLS with intercept term to do linear regression. The output summary() is shown on the next page. There are several terms in output but most key terms which one must focus on are also explained below:

OLS Regression Results

Dep. Variable:	Points	R-squared:	0.476
Model:	OLS	Adj. R-squared:	0.430
Method:	Least Squares	F-statistic:	10.28
Date:	Sat, 31 Dec 2022	Prob (F-statistic):	3.74e-12
Time:	21:52:06	Log-Likelihood:	-623.35
No. Observations:	124	AIC:	1269.
Df Residuals:	113	BIC:	1300.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.2558	16.812	-0.253	0.801	-37.564	29.053
WL	-10.9301	3.884	-2.814	0.006	-18.625	-3.235
Yoga	2.4077	1.381	1.744	0.084	-0.328	5.143
Laps	7.6860	3.507	2.192	0.030	0.738	14.634
WI	0.4366	5.507	0.079	0.937	-10.475	11.348
PAFS	1.4823	1.887	0.786	0.434	-2.256	5.220
Team_Clippers	128.1835	26.319	4.870	0.000	76.040	180.327
Team_Lakers	111.3182	20.283	5.488	0.000	71.134	151.503
Team_Porcupines	124.7833	26.599	4.691	0.000	72.087	177.480
Team_Trailblazers	129.4712	19.234	6.731	0.000	91.364	167.578
Team_Warriors	114.0544	21.725	5.250	0.000	71.013	157.096

Omnibus:	5.894	Durbin-Watson:	1.769
Prob(Omnibus):	0.052	Jarque-Bera (JB):	8.583
Skew:	0.145	Prob(JB):	0.0137
Kurtosis:	4.256	Cond. No.	614.

Notes: [1] Standard Errors assume that the covariance matrix of the errors is correctly specified

coef column

The coef column represents the coefficients for each independent variable along with the intercept value. **Std err is the standard deviation of the corresponding variable's coefficient across all the data points**.

t-values and P>|t|

The t-column provides the t-values corresponding to each independent variable. For example here WL, Yoga, Laps,... all have different t-values as well as different p-values associated with them. **Typically when p-value is less than 0.05, it indicates that the corresponding**

variable is significant. E.g. In the above output, the p-value for all teams are 0. Thus the one hot encoded Team variable is significant. Similarly, the p-value for 'LAPS' is 0.03, and that of 'WL' is 0.006. So, they are also significant variables. Rest of independent variables have p>>>>0.05 and have very less predictive value.

Covariance Type:

Covariance shows how two variables move with respect to each other. If this value is greater than 0, both move in the same direction and if this is less than 0, the variables mode in opposite directions. Covariance is different from correlation. Covariance does not provide the strength of the relationship, only the direction of movement whereas correlation value provides the strength of relationship. Correlation value is normalized and ranges between -1 to +1 and correlation.

DF residuals and DF model

We have a total 124 observations and 11 features. Out of 11 features, 10 features are independent. The DF Model is therefore 10. DF residual is calculated from total observation-DF model-1 which is 124–10–1 = 113 in our case.

R-squared and adjusted R-squared:

It is the percentage of variance in data explained by your model Or measure of how much linear information the predictors contain about the response variable. Currently it is 47.6%. Adjusted R-squared is similar, but it penalizes for adding a non meaningful predictor to the model. So

they are used like this: Suppose with K predictor R-squared is 0.7 and adjusted R-squared is 0.68. Now we add $(k+1)^{th}$ predictor and R-squared increases to 0.75 but adjusted R-squared decreases to 0.64. Then that predictor is **non meaningful for explaining the response variable.**

However, suppose on adding the $(k+1)^{th}$ predictor and R-squared increases to 0.75 and adjusted R-squared also increases to 0.72. Then that predictor is meaningful for explaining the response variable. Also note that, if we obtained OLS without the intercept term then both R-squared and Adjusted R-Squared come with a tag '**uncentered'.** Their values are usually much higher (in our case, both were above 90%). But **they must not be used as a measure of model quality**.

F-statistics

Higher the F-statistics value the higher the impact the predictor variables are having on response variable. For model without intercept, we get the F-statistics **=132.4** but for model with intercept we get F-statistics= **11.69** which is average.

Prob(F-statistics): lower the probability of F-Statistics better the model

Here for the model without intercept, we get a value of Prob(F-statistic)=**$6.63e^{-58}$** but for the model with intercept we get Prob (F-statistic)=**$3.74e^{-12}$**. Once again, the values of F-Statistics and Prob (F-statistic) for model without intercept are better but **they must not be used as a measure of model quality**.

Durbin-Watson Test:

DB test is a measure of homoscedasticity (constancy of variance) in our data. This is an important assumption of linear regression.If its value is in range [0,2], it is usually acceptable. **Ideally it must be in range [1,2]**. In our case, we got a value of 1.769 which is just OK. If DBT value is in range [2,4] then homoscedasticity is violated and we need to do something to data. The DB test is also a measure of autocorrelation.

Omnibus and Jarque-Bera (JB) Test:

Both Omnibus and JB tests are measures of normality in our data. Data is normally distributed is one important assumption of linear regression. **For perfectly normally distributed data, Omnibus and JB are 0 and**

Prob(Omnibus) and Prob(JB) are 1. In our case, for a model with intercept, **we got Omnibus=5.984, far bigger than 0 and prob(Omnibus) is only 0.052 so we can say that normality assumption is violated** and perhaps we should consider variable transformation or use non-linear models.

Skew:

Skew is a measurement of symmetry in our data, **Skew near 0, indicates a perfect symmetry and Skew=0, normal distribution**. Here we get 0.145 indicating data distribution is not perfectly symmetric or normal.

Kurtosis:

Kurtosis measures the peakiness of our data, or its concentration around 0 in a normal curve. **Higher kurtosis implies fewer outliers**. High kurtosis indicates the distribution is too narrow and low kurtosis indicates the distribution is too flat. **A kurtosis value between -2 and +2 is good to prove normalcy**. Here we get Kurtosis =4.256, indicating that the normality **assumption is violated.**

Condition Number:

Condition number is a measurement of the sensitivity of our model as compared to the change of size of the data it is analyzing. High condition number indicates that there is possible multicollinearity present in the dataset. The 614 value in above output is quite high.

AIC and BIC

AIC stands for Akaike's Information Criteria and it provides an estimate of prediction error. Given a data-set and collection of models, AIC is helpful to know which model is the best i.e. with the least prediction error. [**AIC however can not be used to compare two time-series models].** The roots of AIC lie in 'Information Theory' which says that if a data-set has information $\mathbf{I_D}$ and if a model captures information **I** then **I** is always less than $\mathbf{I_D}$ **(**no model can capture all the information from data**). AIC is a measure of the fraction of information gained by model from total information in data. So higher AIC is better.** However there is much more to it. As we go on increasing model complexity, the information gain goes on increasing but now AIC starts penalizing the model thus preventing overfitting. **Thus on one side AIC allows to increase information gain (prevent underfitting) , on the other**

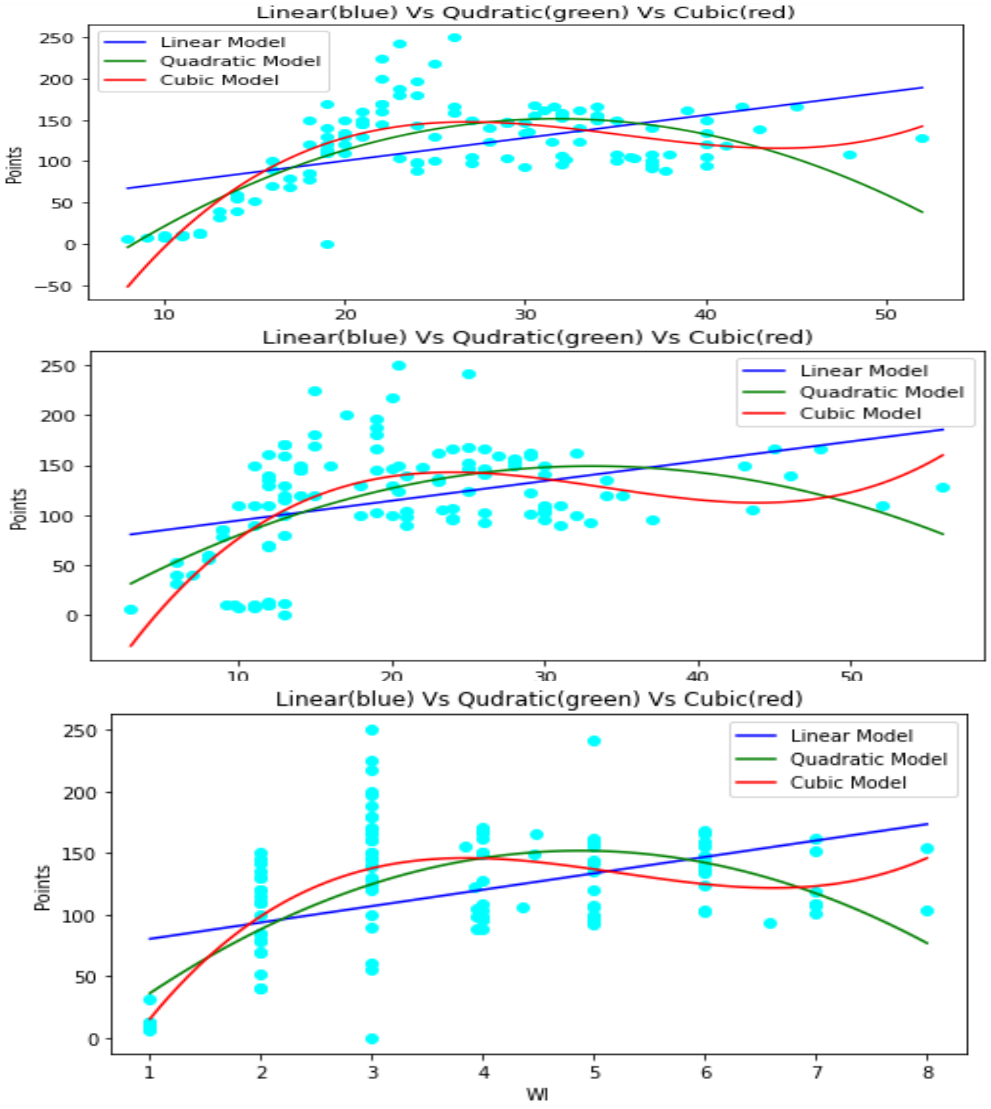
side it also protects against too much complexity (prevents overfitting).
The formula for AIC is $AIC = 2k - 2ln(L)$. Where,
k=no of predictors , **L= Max Likelihood function**.
Note: For small datasets (N<40), we have to use corrected AIC (AICC).
BIC stands for Bayesian Information Criteria **and is also used as criteria for model robustness**.

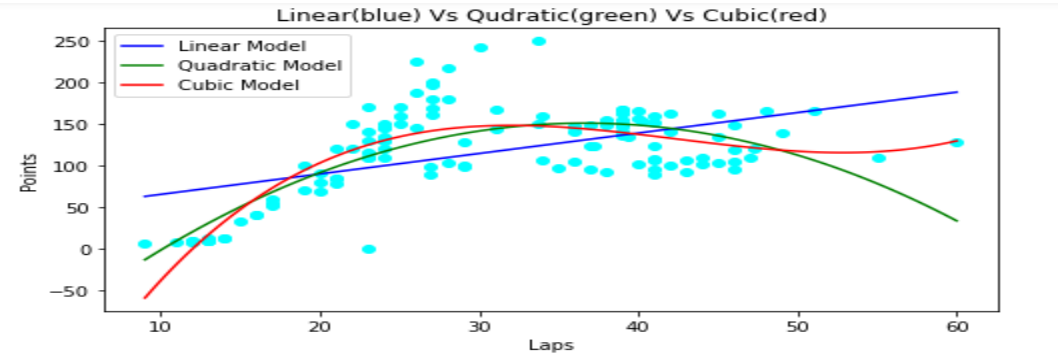
Log-Likelihood:
The log-likelihood value is a measure for fit of the model with the given data. It is useful when we compare two or more models. T**he higher the value of log-likelihood, the better the model fits the given data.** It can range from negative infinity to positive infinity.
NOTE: Actually likelihood and probability are totally different statistical concepts. We avoid their discussion here but readers are recommended to these links: link-1, link-2 or link-3.

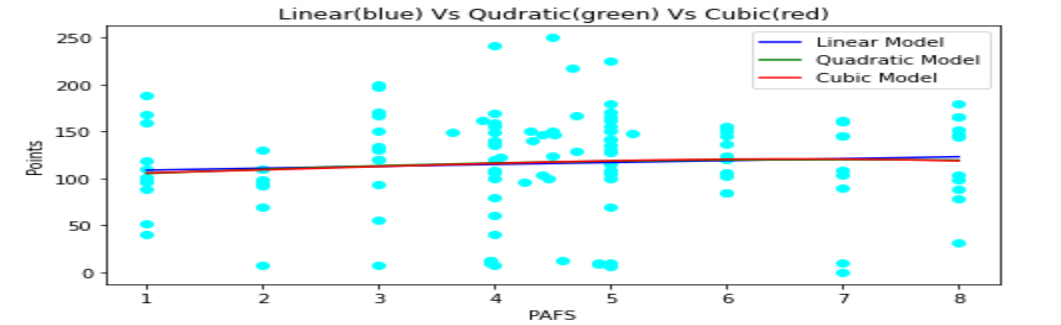
The above output of the OLS model was on the train set. It already shows that R-squared and adjusted R-squared is not very good. Still we used the model to make predictions on the test set and found three key metrics MAE, MSE, RMSE. We got **MAE = 32.82**, which means the predictions of various points are 32.82 units above/below the actual

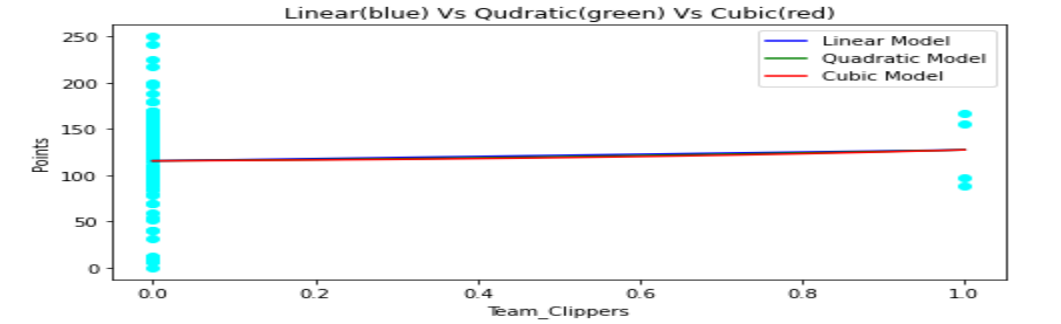
value on avg. We got **MSE=1782.34** which is a high value and **RMSE=42.22** which means out of 100 data-points that we are predicting, error is associated with nearly 42 points. This is too high a rate of error and unacceptable. The value of R-squared is 0.4633 which is also low.

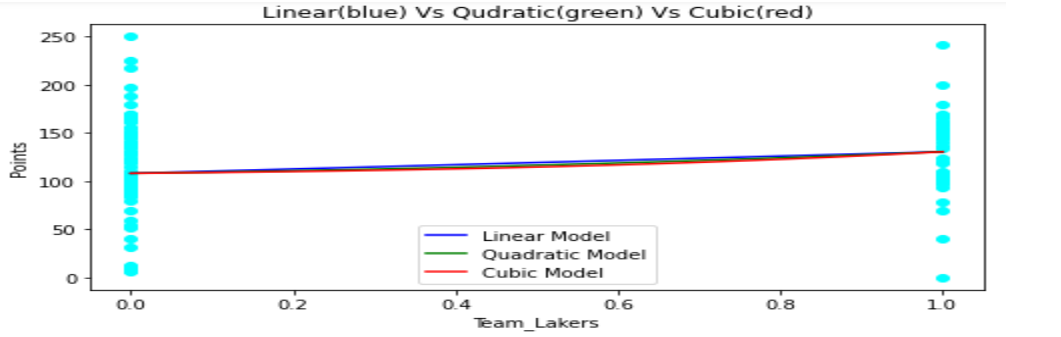
Finally we used the Sklearn library to do polynomial regression. We got the following plots for ponts Vs predictors by linear, quadratic and cubic curves. So, the quadratic curve looks best fit from below plots while cubic plots show overfitting. **Also from plots for PAFS and various 'Teams' we simply removed those variables**.

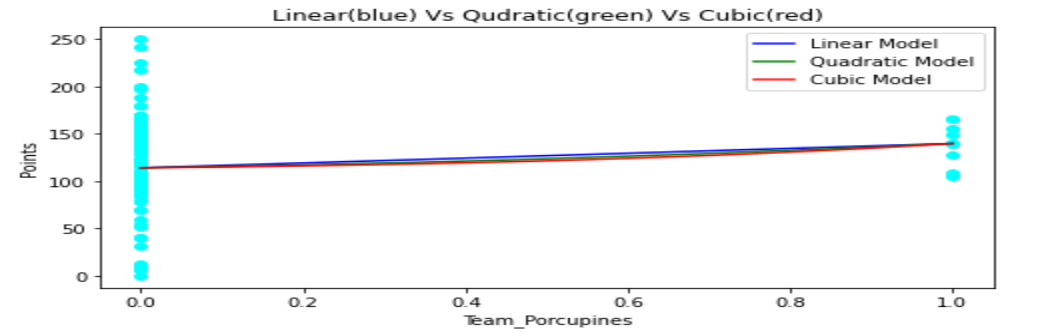


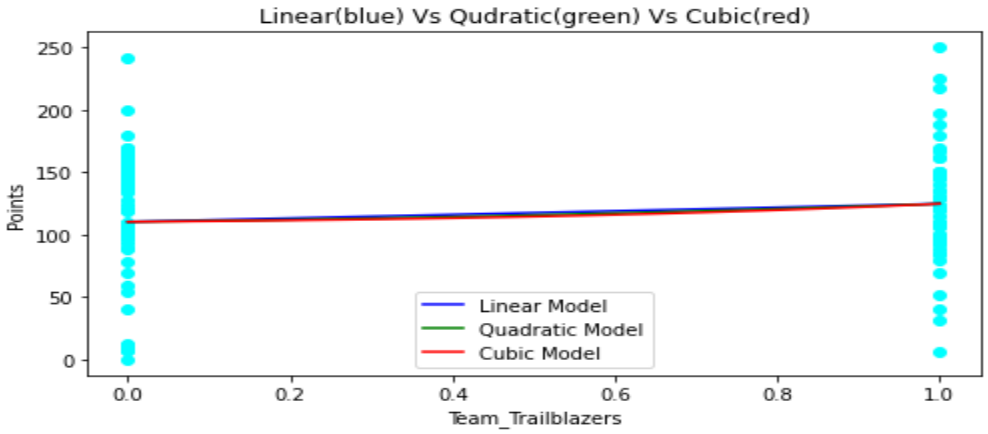

Linear(blue) Vs Qudratic(green) Vs Cubic(red)
Linear Model
Quadratic Model
Cubic Model
Points
Laps

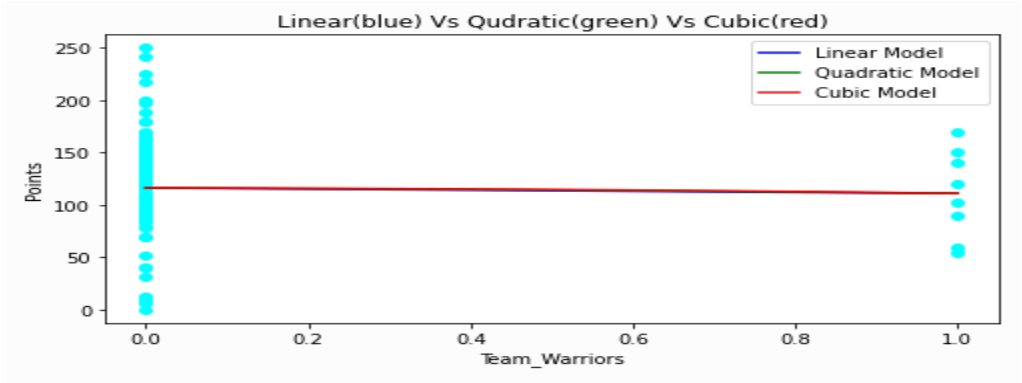

Linear(blue) Vs Qudratic(green) Vs Cubic(red)
Linear Model
Quadratic Model
Cubic Model
Points
PAFS


Linear(blue) Vs Qudratic(green) Vs Cubic(red)
Linear Model
Quadratic Model
Cubic Model
Points
Team_Clippers


Linear(blue) Vs Qudratic(green) Vs Cubic(red)
Linear Model
Quadratic Model
Cubic Model
Points
Team_Lakers


Linear(blue) Vs Qudratic(green) Vs Cubic(red)
Linear Model
Quadratic Model
Cubic Model
Points
Team_Porcupines





Finally we got **R-squared=0.7252** (<u>improved 25% compared to linear regression</u>) and **RMSE=30.21** (<u>Decreased 12% compared to linear model</u>). Thus the quadratic model is far better but still the question is if such huge improvement was a result of using quadratic curve or result of removings columns for Teams and PAFS. In fact, applying the same quadratic curve on a data set with all columns gives a highly overfitting model. There are also questions like:

A. Why even nonlinear correlation coefficients (**CCC**) are not that strong and what to do in such cases?

B. What happened to Anova's result that 'Team' significantly affects the 'Points'? (If we keep all columns the model grossly overfits.) **See code-3**

To conclude, an 'innocent' looking small dataset is still giving us trouble. Seems the story is not over yet! We will continue in the next post with regression splines and then another post on regularization in regression.