**About:**

The Indian Premier League is a professional Twenty20 cricket league, contested by eight teams based out of eight different Indian cities.

A team can acquire players through any of the three ways:

* The annual player auction
* Trading players with other teams during the trading windows, and
* Signing replacements for unavailable players.

Players sign up for the auction and also set their base price, and are bought by the franchise that bids the highest for them. Unsold players at the auction are eligible to be signed up as replacement signings. In the trading windows, a player can only be traded with his consent, with the franchise paying the difference if any between the old and new contracts. If the new contract is worth more than the older one, the difference is shared between the player and the franchise selling the player.

There are generally three trading windows, two before the auction and one after the auction but before the start of the tournament. Players cannot be traded outside the trading windows or during the tournament, whereas replacements can be signed before or during the tournament.

Some of the team composition rules are as follows:

* The squad strength must be between 18 and 25 players, with a maximum of 8 overseas players.
* [Salary cap](https://en.wikipedia.org/wiki/Salary_cap) of the entire squad must not exceed ₹850 million (12 million US dollars).
* Under-19 players cannot be picked unless they have previously played [first-class](https://en.wikipedia.org/wiki/First-class_cricket) or [List A](https://en.wikipedia.org/wiki/List_A_cricket) cricket.
* A team can play a maximum of 4 overseas players in their playing eleven.

The term of a player contract is one year, with the franchise having the option to extend the contract by one or two years.

**Problem Statement:**

The idea of the project is to identify and predict the value of each IPL player, based on certain obvious features that significantly affect the bid value of the player in the IPL group.

This, prediction of the value will also give an idea of what the maximum value a player could be asked for as a bid, rather than bidding a considerably higher amount all on just one player and then running short while getting to choose the other players for the team.

**Approach:**

I read all the IPL data from the year 2008 until 2021 and created a merged data frame of all IPL data. The data consists only of the features that contribute to the value of a player such as the runs above average, EF score, the wins and their take home salary.

**About the variables:**

1. Runs Above Average:

The batsman ranking is calculated based on three attributes namely, the total runs scored, the total balls faced and the total outs. The idea is to capture all if these three attributes in a single number called Runs above average.

So, an RAA has two parts, the first part captures the strike rate of the batsman and the second part captures the rate at which the batsman gets out.

RAA = (Batsman runs – Average batting strike rate \* Balls faced by batsman ) + Overall batting average \* Balls faced by batsman \* (Average out rate – Batsman out rate)

1. Wins:

The idea behind this attribute is that, there are 10 wickets that the opposition needs to take for the team to win the match. Thus, the value of 10, “average” batsmen is equal to one win.

Wins = RAA / (10 \* Overall batting average)

1. EF-Score:

It is the Eigen Factor Score. This is a metric, that is used to rank the teams in different formats of the game. This is a new way of ranking the teams in ICC, by considering the relative strengths of the teams.

So, a victory against a relatively stronger team will give a higher EF-Score, compared to a victory against a weaker team. The advantage of this method is that, it’s a non-parametric way of approach, by not giving any additional parameters for computing the team rankings.

The dataset consists of both missing values and different data types. Thus, it is always important to perform EDA on the data before constructing a regression model to predict the value of the player.

The datatypes of all the features are different. Some are considered as objects since the salary and the value column of the player contains dollar symbol and commas in between. So, after replacing these special characters with nothing, I then convert all of the data types to float before proceeding to model building.

Then I use some visualization on the all the features to see if the data is normally distributed and also remove the columns that do not add value to the prediction of the player value such the name of the player and the team in which he has played.

All that is left is to construct a model. There are actually numerous regression models that can be used, for which I have used an inbuilt library called the Pycaret. This problem of predicting the value of the IPL player is a regression problem, since the output variable that is to be predicted is a continuous variable, which cannot be classified.

This library helps in conserving time of the coder, by fitting the entire data, performing all feature engineering and constructing all the possible regression models available on the dataset, by just giving the input data and the target variable. PyCaret is an open source, low-code machine learning library in Python that allows to go from preparing the data to deploying your model within minutes in my own choice of the notebook environment.

For better accuracies, the hyperparameter of the library could be tuned further. Some of the regression models that Pycaret calculates are:

* Cat Boost Regressor
* Gradient Boosting Regressor
* Lasso Regression
* Ridge Regression
* Least Angle Regression
* Lasso Least Angle Regression
* Bayesian Ridge
* Linear Regression
* Light Gradient Boosting Machine
* Huber Regressor
* Orthogonal Matching Pursuit
* Random Forest Regressor
* Extra Trees Regressor
* Passive Aggressive Regressor
* AdaBoost Regressor
* Extreme Gradient Boosting
* K Neighbours Regressor
* Elastic Net
* Decision Tree Regressor

This library also gives the loss functions of all the model that is fitted. A loss function gives the error difference between th actual value and the predicted value. Some of the loss function that Pycaret regression library returns are:

* MAE
* MSE
* RMSE
* R2
* RMSLE
* MAPE
* TT

Thus, after fitting the input data to the pycaret library, the best model with relatively higher accuracies and lower loss function of RMSE is the Cat boost Regressor. The metric and the loss function depend on the requirement of the problem that should be solved. Here, I consider the accuracy and RMSE values to choose the best model. So, I check what are the initial parameters that pycaret has taken to run the cat boost regressor technique and then start tuning the parameters of the cat boost technique.

Moreover, a cat boost regressor model does the following:

Cat boosting technique is built upon the decision trees and gradient boosting. The main idea of a boosting technique is to sequentially combine many weak models and thus through a greedy search make a competitive model with higher accuracy of prediction. This is because, once the decision trees are connected sequentially, they learn from the mistakes of the earlier trees and tune the parameters to reduce the errors. This process of adding a new function to existing ones continues until the selected loss function is no longer minimized.

Unlike other gradient boosting models, cat boosting technique grows oblivious trees. This means that the trees are grown by imposing the rule that all the nodes at the same level, test on the same predictors, with the same condition. So, the index of the leaf can be calculated using a bitwise operator. It promotes efficiency of the CPU and on the other hand, it also tries to optimize the solution and avoid overfitting with a minimal error.

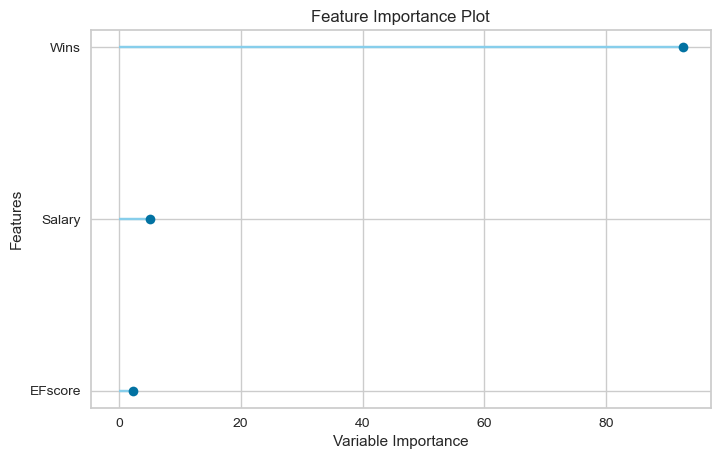
Now the accuracy of the model for predicting the value of the IPL player is 89.5%. In order to deploy the project using Flask and Heroku app, I have created a pickle file of the model in the jupyter notebook.

Pickling is useful for applications where I need some degree of persistency in the data. The program's state data can be saved to disk, so that I can continue working on it later on. It can also be used to send data over a Transmission Control Protocol (TCP) or socket connection, or to store python objects in a database. Pickle is very useful for when working with machine learning algorithms, where it is required to save them and be able to make new predictions at a later time, without having to rewrite everything or train the model all over again.

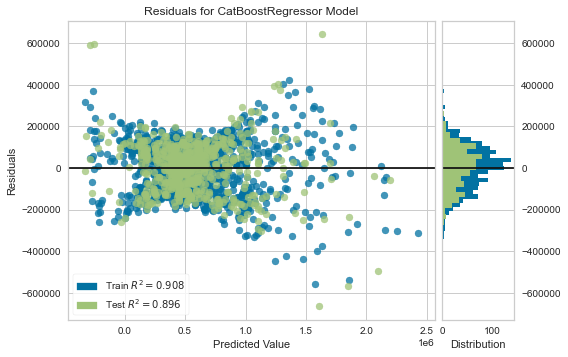
**Visualization:**

Some visualisations that helps get a better understanding on the data are as follows:

1. The important features that significantly contribute to the prediction of the player value is given by the graph below:



1. The residual errors in the data points by comparing the test and train data:



1. The QQ Plot, which is the probability plot, which is a graphical method that compares two probability distributions by plotting their quantiles against each other. If both the sets of the quantiles come from the same distribution (say both are from normal distribution), we will be seeing that points roughly form a straight line.



**Deployment:**

Initially there is a main python file that connects to both html, css and the pickle file. I have created two html pages, one for the home page that gets values for the user for the features RAA, EF Score, Wins and Salary. The other html page is the appearance of the predicting page once the user enters all the necessary features into the input or home page.

The designing and the way that both input and output page should appear is set by the chose in css.

Now, I have imported the model details through the pickle file where I had dumped the model into from the jupyter notebook. Once all this is set, I had deployed the project through flask. However only Heroku app gives a website that can be access by the common people. In order to deploy the model on Heroku, I have created a Procfile and a requirements text document in visual studio code and created a Heroku account, linked it to GitHub and finally deploy it.

**Reference:**

<http://www.cricmetric.com/blog/glossary/>

<http://www.cricmetric.com/blog/2012/02/a-context-independent-method-of-ranking-odi-players/>