**Project Title: Machine Learning Model Deployment with IBM Cloud Watson Studio**

**Phase 2 : Innovation**

Real-time Problem :

Predict the total score (runs) a team is likely to achieve in an IPL match based on factors such as venue, pitch conditions, team composition, and historical performance.

**Technology used :**

* Machine Learning.
* Deep Learning.
* Flask (Front-end integration).

Well, for the smooth running of the project we’ve used few libraries like NumPy, Pandas, Scikit-learn, TensorFlow, and Matplotlib.

**Step-by-step implementation:**

### 

### **Step 1: Understanding the dataset!**

### 

### Cricsheet is considered as an appropriate platform for gathering the data thus weare taking the data from <https://cricsheet.org/downloads/ipl.zip>. It contains data from the year 2007 to 2021. IPL players’ state has been taken from data world. The dataset contains details of every IPL player from the year 2016 – 2019.

### **Step 2: Data cleaning and formatting**

### ****The quality of your data has a significant impact on the accuracy and performance of your machine learning model**.**

### Here imported both the datasets using *read\_csv()*method into a dataframe using pandas and displayed the first 5 rows of each dataset. We did some changes to our dataset like added a new column named “y” which had the runs scored in the first 6 overs from that particular inning.

### **Step 3: Encoding the categorical data to numerical values.**

### For the columns to be able to assist the model in the prediction, the values should make some sense to the computers we may choose to do the process manually, the **Scikit-learn library** gives us an option to use ****LabelEncoder. Label encoding involves converting each category in a categorical variable into a unique integer. This method is suitable when there is a natural order or ranking among the categories. In Python, we can use libraries like scikit-learn or pandas to perform label encoding.****

### **Step 4: Feature Engineering and Selection**

### ****They involve creating meaningful features from the existing data and choosing the most relevant features for your model.**** We will then divide our data into train sets and test set before using a machine learning algorithm.  it is always a better choice to scale your data before processing it. Here we can use **MinMaxScaler**from **sklearn.preprocessing**

### **Step 5: Building, Training & Testing the Model**

### ****Start by splitting the dataset into three parts: a training set, a validation set, and a test set. An appropriate machine learning algorithm or model for the problem is choosen.****

### To train the model using the training data we will import ****Sequential****from  ****tensorflow.keras.models.****

### Also, we will import ****Dense & Dropout****from ****tensorflow.keras.layers****as we will be using multiple layers.

### ****Once we are satisfied with the model's performance on the validation set, we can evaluate it on the test set to get an unbiased estimate of its generalization performance.****

### **Step 6: Predictions!**

* + - **we will be predicting our X\_test. Then we will create a dataframe that would show us the actual values and the predicted values.**
    - **when our model is predicting quite well. and giving us almost similar scores. To find out more accurately the difference between actual and predicted scores, performance metrics will show us the error rate using mean\_absolute\_error and mean\_squared\_error from sklearn.metrics**

* + - **Continue to monitor the performance of your deployed model and update it as necessary. Over time, data distributions can change, and your model may become less effective.**