Sports Analytics for Major Tennis Tournaments:

- I was a subject matter expert and data science consultant in the project.
- Project objective was to generate the highlight of major tennis matches using Hawkeye data.
- It's classification problem on a supervised dataset with label value as : shot is a winner or not.
- **Major features we have used here are:**
  1. Distance of ball bounce from corners, baseline, and net.
  2. Relative horizontal distance (X1-X2) of player.
  3. Speed of ball on bounce (so basically the data was in arc format with 3d coordinates of the ball and player at the time of hit) – we have generated new features.

  On proper analyzing we got to know that the given dataset was an imbalanced dataset with most the labels as "Not a winner"

  So, using SMOTE oversampling technique we were able to balance the minority and majority class and had applied the classification algorithms random forest to predict the best shots and serves with an accuracy score for 90% .
  Note: from imblearn.oversampling import SMOTE

  Dataset size: 2 million (9000 was winners) Men's Quarterfinals to Finals of last 5 years.

  Performance Matrix:
  Logistic Regression, Random Forest, and SVC.

  Hyperparameters:
  1. **Random Forest Classifier** was giving the best accuracy of 88% :

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini'
,
            max_depth=80, max_features=2, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=8,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

  It's been a long time so it's hard to recall the exact values but as per my knowledge the values of the hyperparamter are:

  N_estimator = 100, Max_features = 2, Max_depth =80, Min Samples leaf =5
  min_samples_split =8

2. Support Vector Classifier:

param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001]}

```
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

How to deploy ML model.

We were saving the model using **dump() function** of pickle libraray.

```
pickle.dump(model, open('model.pkl', 'wb'))
```

This will serialize the object and convert it into a "byte stream" that we can save as a file called model.pkl.

To load a saved model from a Pickle file, all you need to do is pass the "pickled" model into the Pickle **load() function** and it will be deserialized. By assigning this back to a model object, you can then run your original model's predict() function, pass in some test data and get back an array of predictions.

```
pickled_model = pickle.load(open('model.pkl', 'rb'))

pickled_model.predict(X_test)
```