**Data Science Workshop - NBA Free Throws Prediction**

**Phase I: Choose a dataset from Kaggle**

In January 2015 a [data-set](https://www.kaggle.com/sebastianmantey/nba-free-throws) of 600K NBA free-throws was upload to Kaggle by Sebastian-Mantey. The data was scraped from the website [ESPN.com](https://www.espn.com/) which belongs to an entertainment and sports programming network.

Dataset contains 11 variables:

* end\_result: host total score - guest total score
* game: host team vs guest team
* game\_id: id of specific game
* period: which quarter
* play: who make free throw, make or miss free throw
* player: player name
* playoffs: whether a playoff game or regular game
* score: host team score - guest team score at that time
* season: NBA season
* shot\_made: whether player got the free throw
* time: time left in that quarter

Number of free-throws: 618,019 in which 575893 regular games and 42126 playoffs. Number of unique games: 12,874.

**collecting more data**

In order to expand our dataset, we decided to use an open source python library PandasBasketball, and use a web scrapper in order to get more players stats from [https://www.basketball-reference.com](https://www.basketball-reference.com/) website.

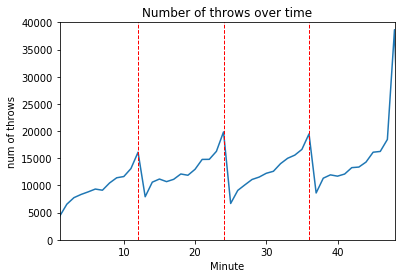
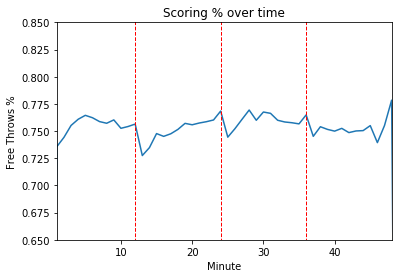
for each player we added the following columns:

* Position: The most common position for the player over his seasons.
* FG%: The player's shooting percentage throughout his career
* 3P%: The player's 3-points throws percentage throughout his career
* FT%: The player's free-throws percentage throughout his career
* Height: The player's height
* Weight: The player's weight
* ShootingHand: The player's shooting hand
* draftRank: The player's draft-rank

Some players stats had been inserted manually because some bugs found on PandasBasketball library.

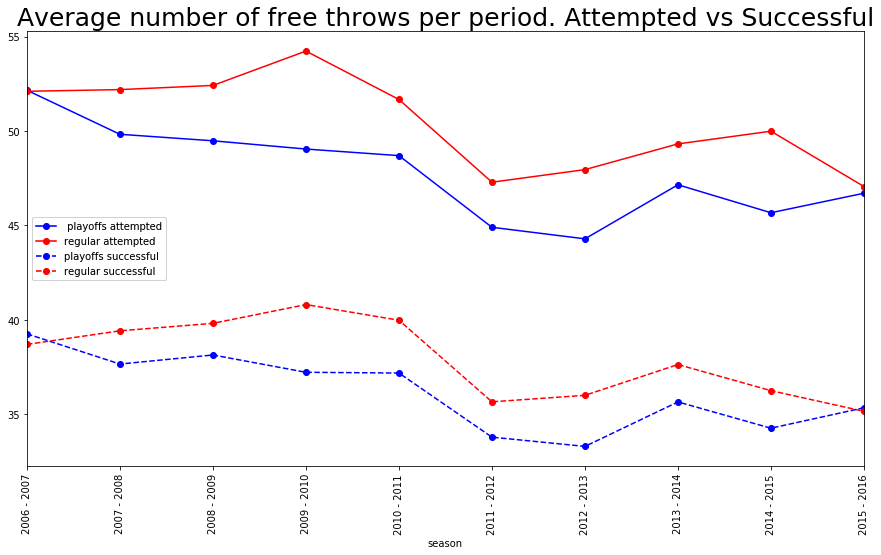
**Phases II: Analyze the dataset to understand its nature and properties**

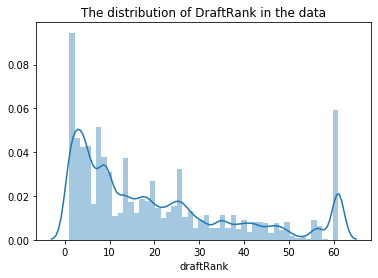
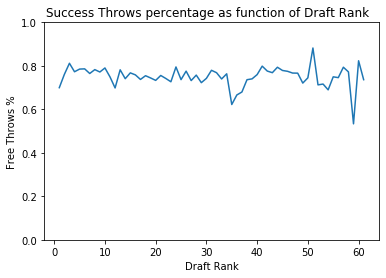
In order to understand our data, we created some graphs that show the distribution and correlation of our features.

We first wanted to see the distribution of the throws throughout the game time. In the original dataset, the time column represents the time left in that quarter and the period column represents the quarter of the game, so we added a new column to our data which calculate the absolute time in the game that the throw was made.

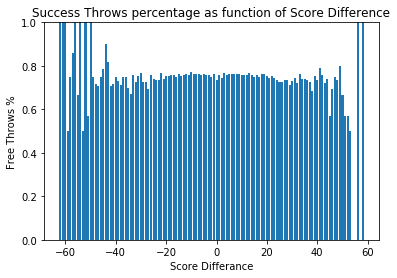
In the graphs, we can observe that at the beginning of every quarter, both the number of free throws and the success percentage drops. Moreover, at the end of a quarter, and especially at the end of the game, both plots increase. We can explain this behavior, as in basketball rules, when a team is making more than 5 fouls, every extra foul made in that quarter will be penalty with a free throw.

Later, we wanted to see if there are differences in the number of attempted and success throws between the different seasons and between playoff games and regular season games.

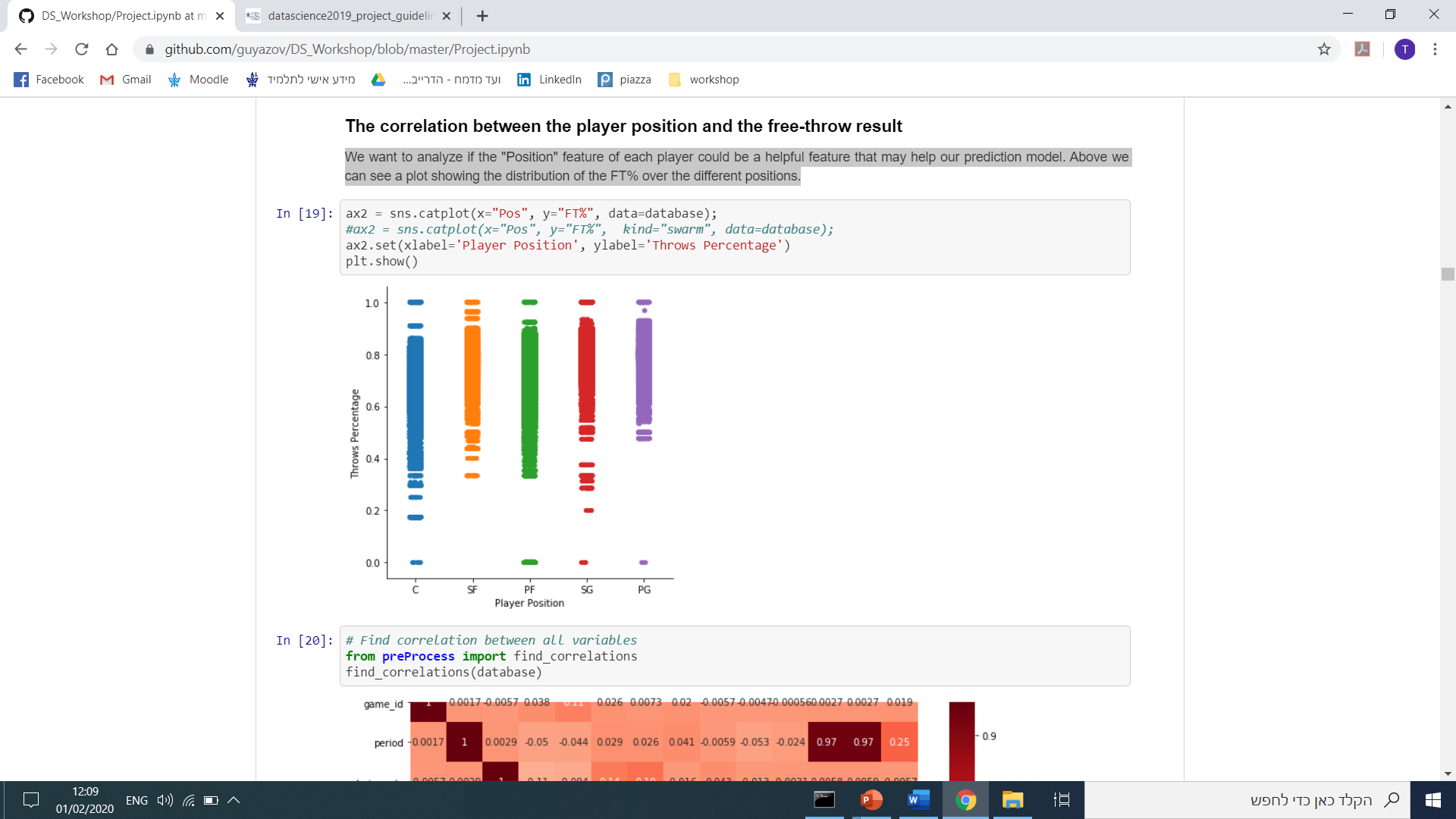
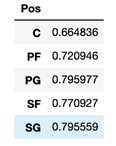
We saw there is no big differences between the seasons at both attempted and successful throws, but there is a clear difference between playoffs and regular season.

Next, we checked the correlation between the player's draft rank and his free-throw percentage. We found that the feature ‘draftRank’ has missing data. We checked manually and discovered that these values are missing because the players performed the free throw didn't have a draft rank because they were not chosen to the draft, so we replaced the cells of "undrafted" and NAN with the score 61 as that player was chosen last to the draft.

We saw, there is no special trend in the graph (the FT% is between 0.7 to 0.9 for all ranks). We expected that the higher-ranked players would have better performance but we can't conclude it from the data.

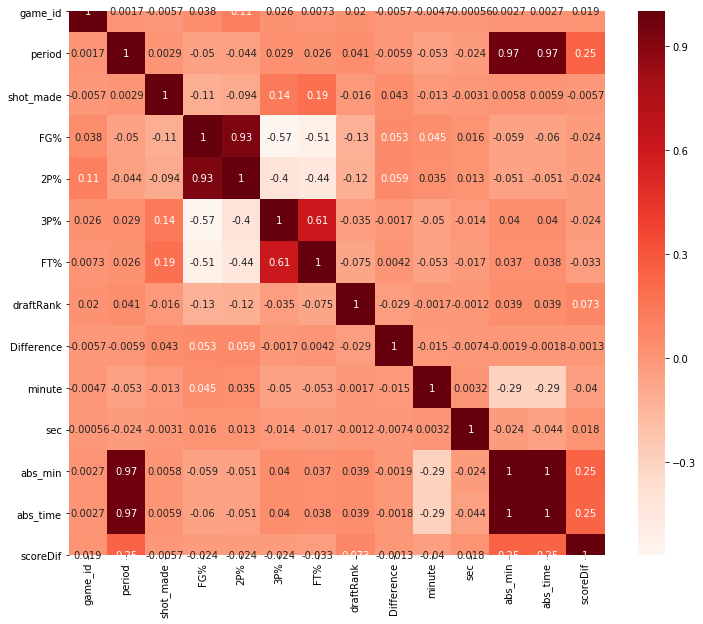
Moreover, we looked at the correlation between the score difference and the free-throw result. We thought that in smaller score difference, the player is tenser and will perform worse. We add a feature calculating the score difference at the time of the throw.

It seems that our hypothesis that as long as the score difference gets bigger, so as the free-throw success percentage is partly true. We can see a trend in the graph but it is not continuous.

Another correlation we checked, is between the player position and the free-throw result.

We saw that every position has different mean and variance. The Centers players owns the worst percentage. Point Guards and Shouting Guards players holds the best percentage over all the positions.

Finally, we checked the correlation between each tow features in our data



**Phase III: Define a data science prediction problem**

We chose to focus on tow prediction problems.

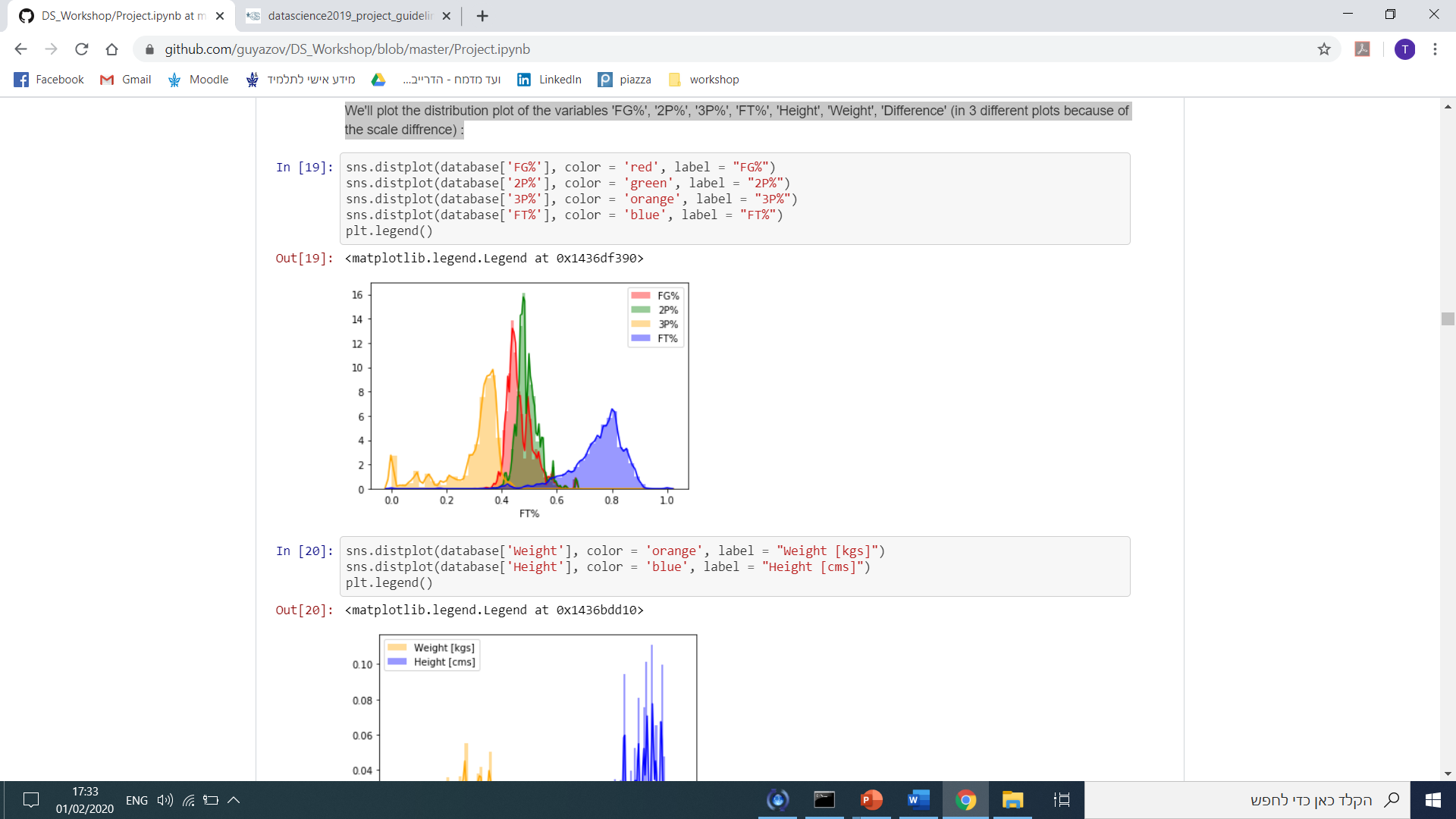
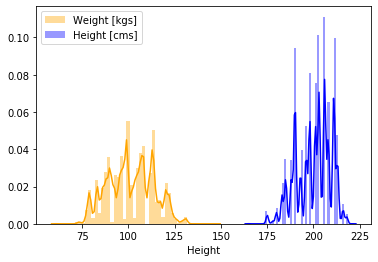
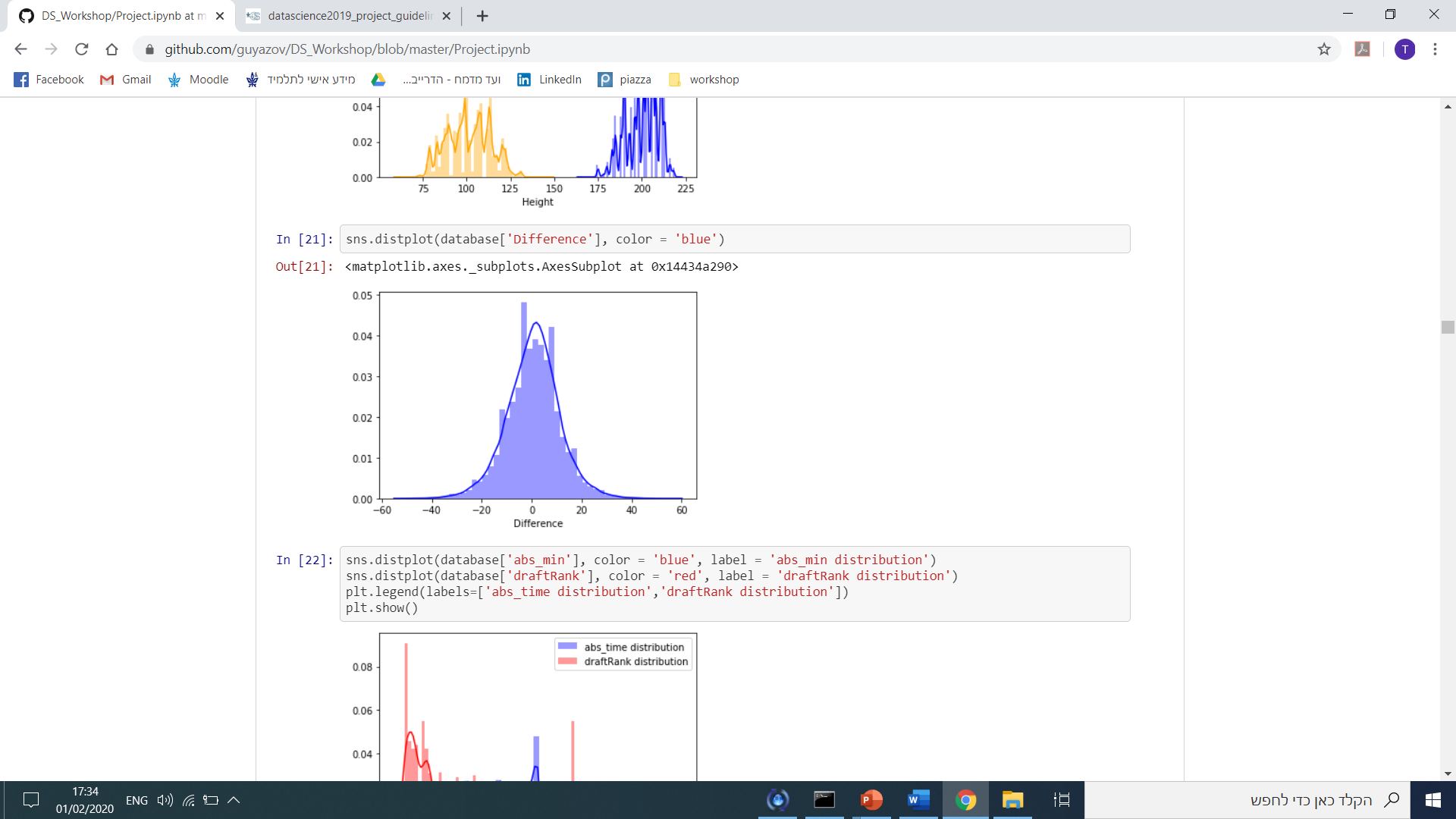
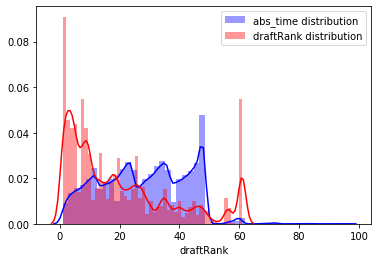
* Main Target: Try to predict whether a player will score a **certain free throw** (online prediction).
* Secondary Target: Predict the players success percentage for the **current** **season** based on his past performance and give the coach an insight about the player (offline prediction).

**Phases IV+V: Sanitize and organize the dataset (Missing data, data integration), Perform feature engineering and extraction.**

We continue analyzing the data and found that the features ‘FT%’ and ‘3P%’ also has missing values since there are players that have never thrown a 3-pointer or free-throws. Since the percentage of the missing data was less than 8%, we filled the cells with the mean of each feature.

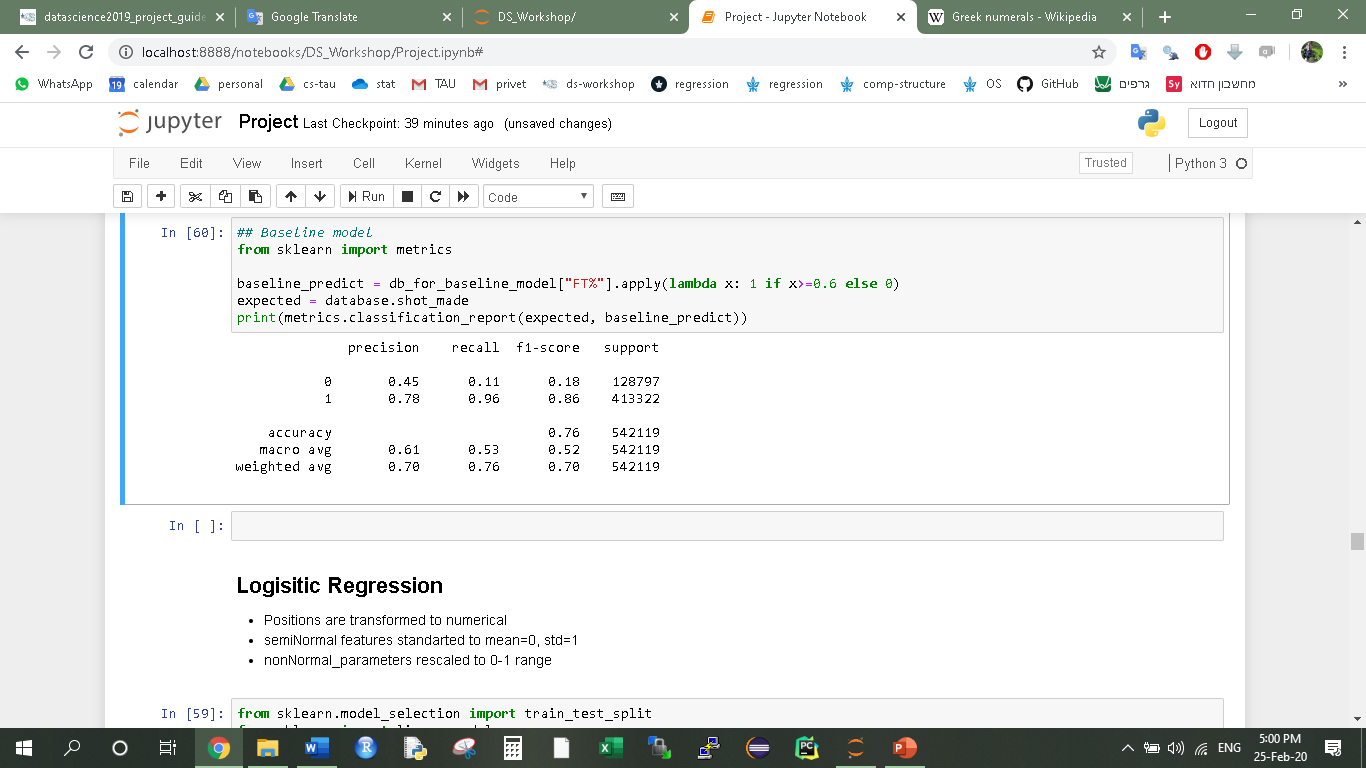
We then changed the categorical variable (position) to numeric and changed the binary and numeric variables to be in the appropriate type.

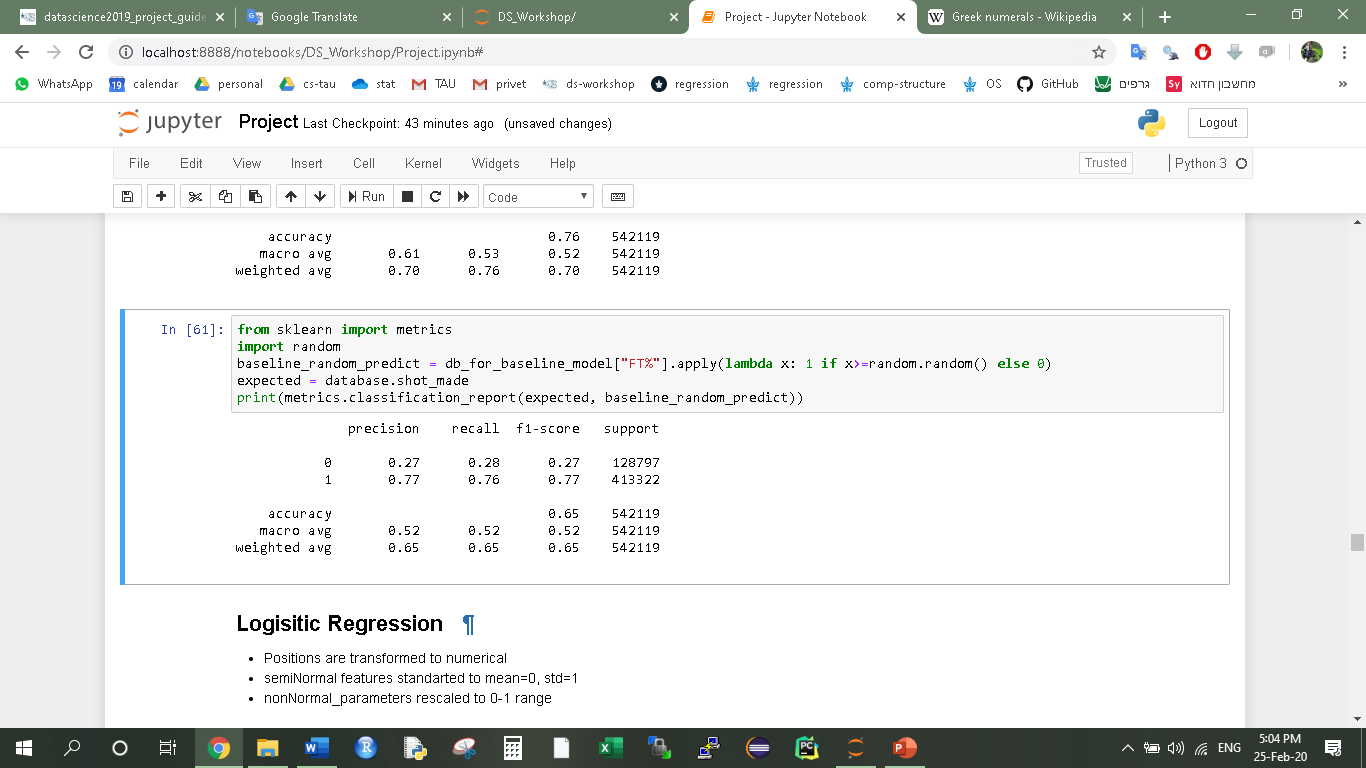
We chose the relevant variables for our model. Our continuous parameters are 'FG%', '2P%', '3P%', 'FT%', 'Height', 'Weight', 'draftRank','abs\_min', 'Difference' and the binary parameters are 'playoffs', 'Pos', 'ShootingHand', 'First\_shot', 'Second\_shot', 'Third\_shot', 'First\_shot\_was\_in', 'Second\_shot\_was\_in'.

We assumed that our variables are normally distributed, and checked this hypothesis and normalize the variables.

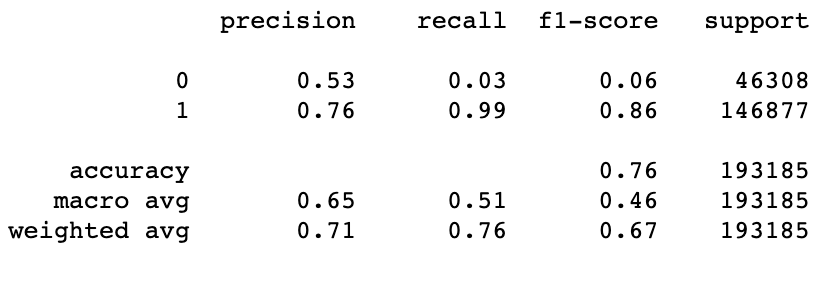
We can see that the continues variables looks like they are semi-normal distributed, so our assumptions are reasonable.

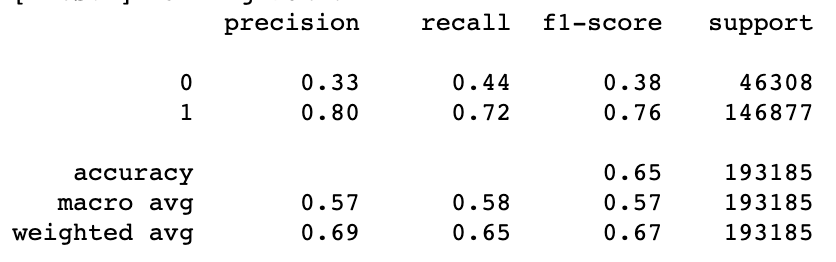
**Phase VI: Implement/Choose appropriate ML algorithms and methods over the data model to output results.**

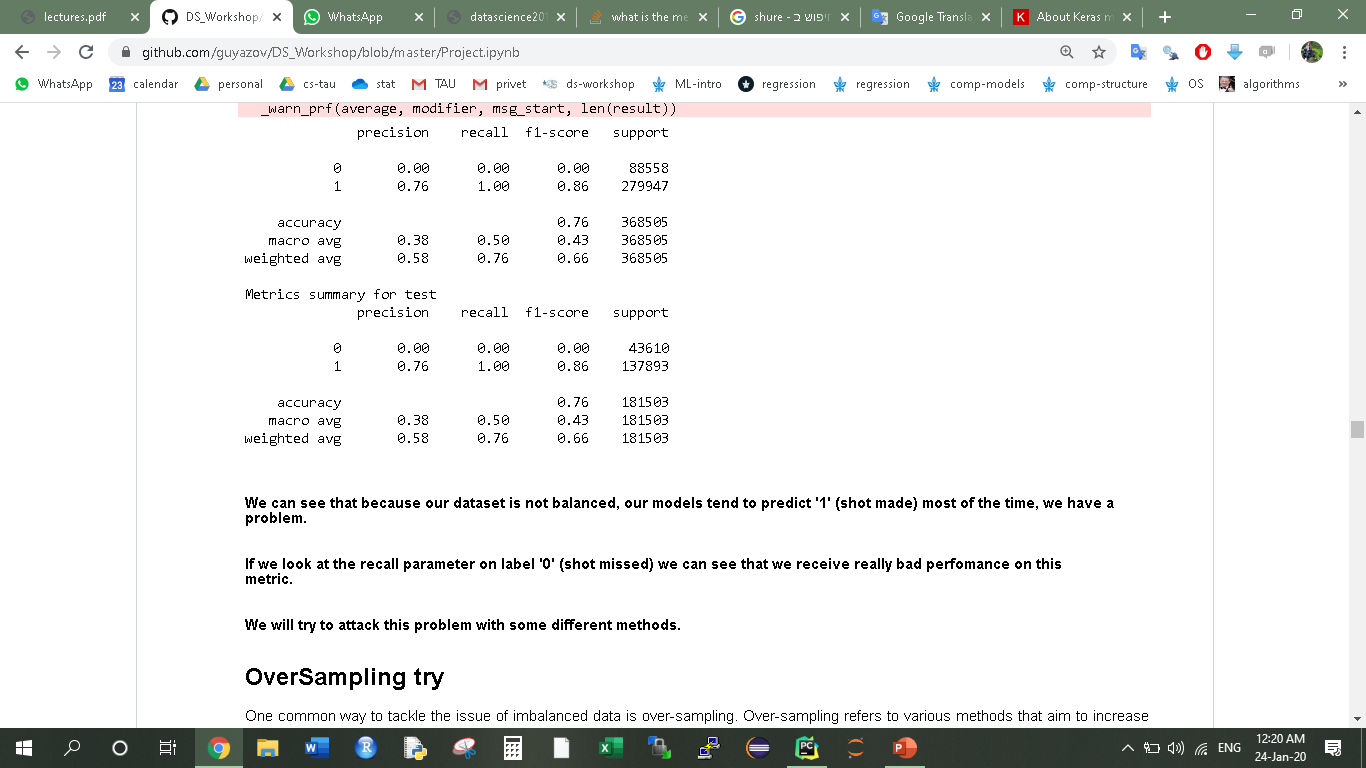
First, we run a baseline model for our prediction problem. We set a rule that each throw made by a player with ‘FT%’ higher than 0.6, the model predicts the shot made and else, the shot missed. We got an accuracy of 72% in the prediction but we saw that the recall on the shot missed is very low.

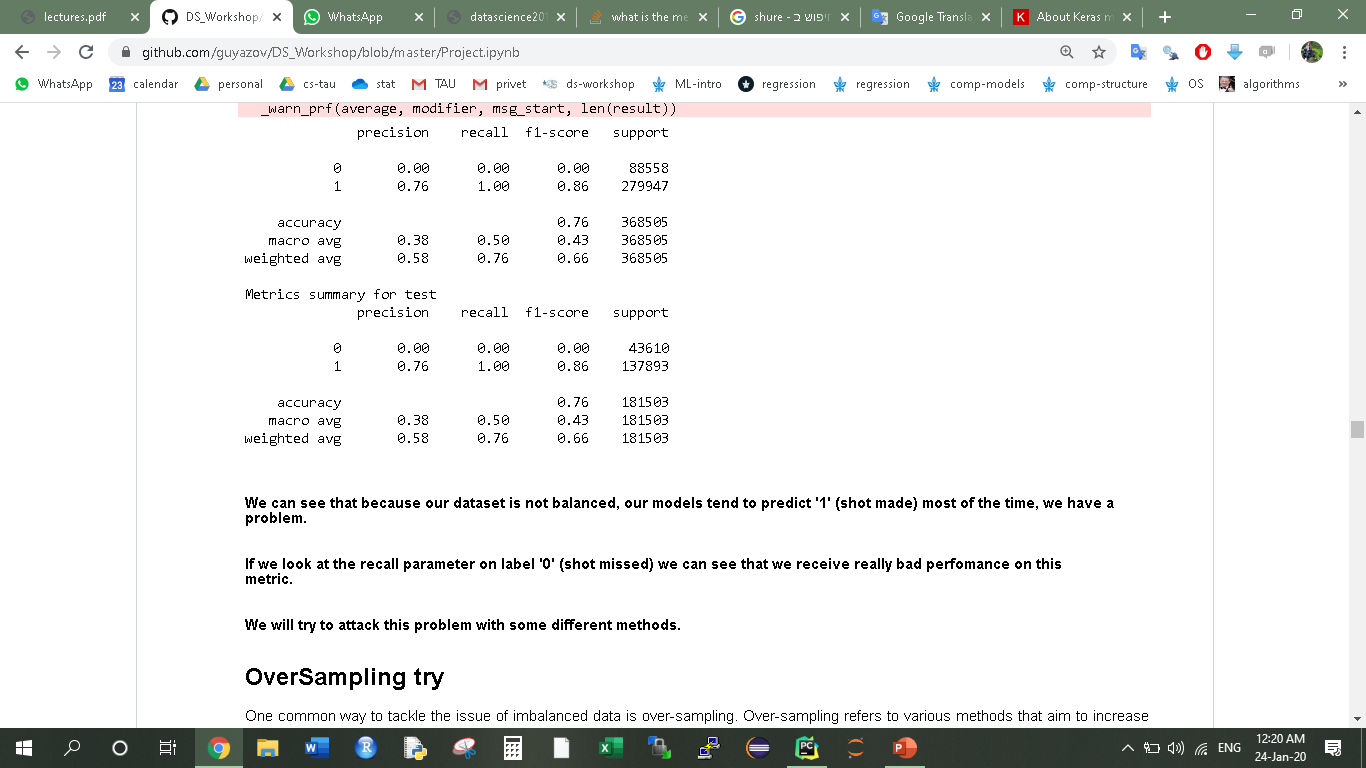
As suggested by prof. Deuch, we tried a random baseline model which toss a coin with the player’s ‘FT%’ score as it’s probability, it performed better on the missed throws recall but the accuracy was worse.

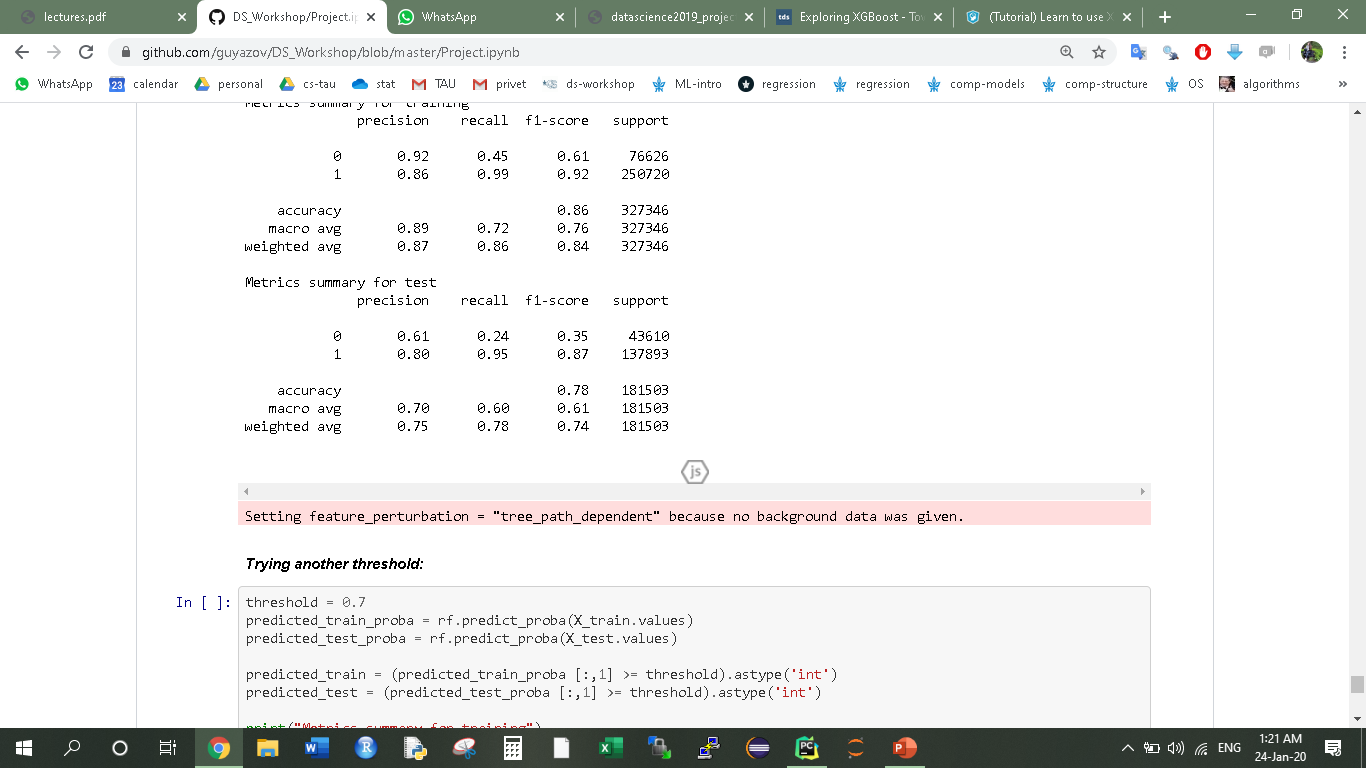
We checked and discovered that our data is **unbalanced**. 76% of the throws with shot made and 24% with shot missed. We split our data to train and test randomly and made sure we keep this ratio in the train and test groups.

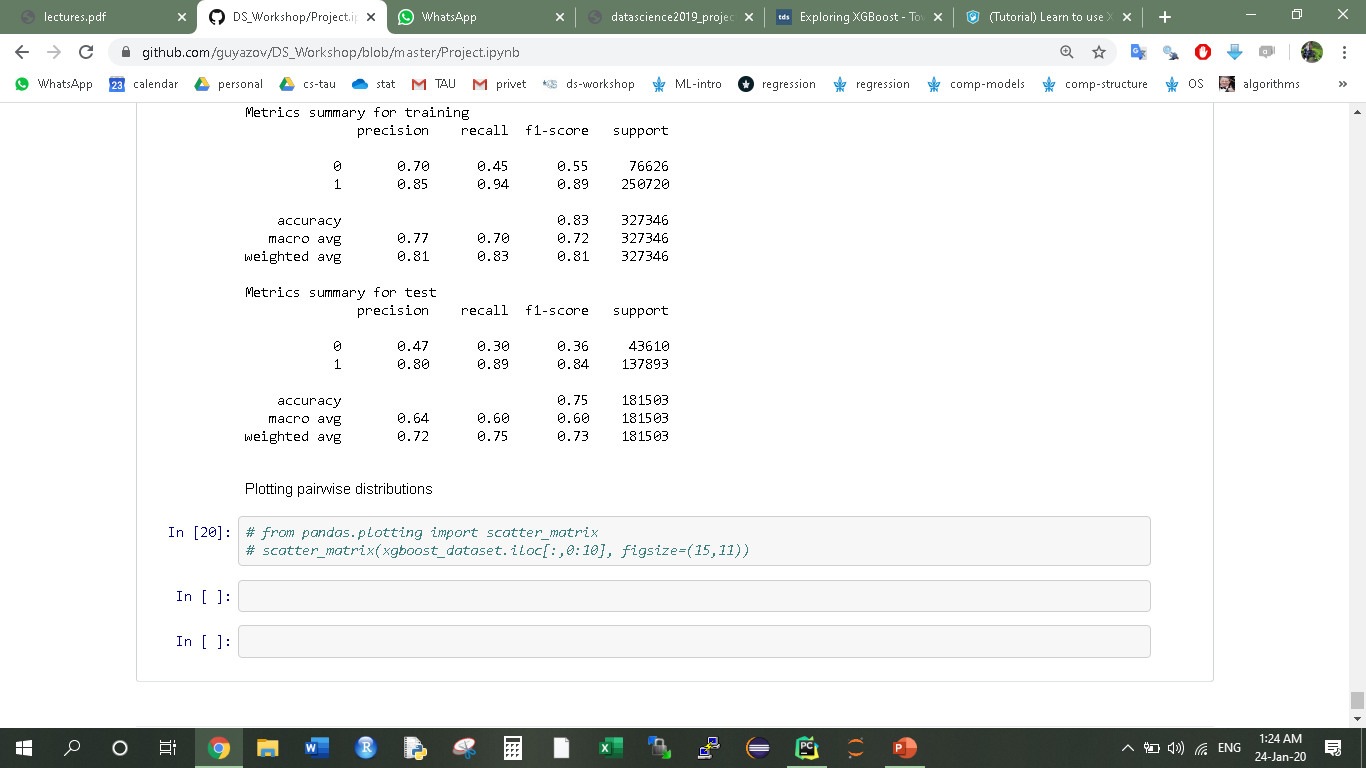
We tried to find more advanced ML models to predict the result of the throw. We used the Logistic-regression model with and got these results on the test:

Then we tried the SVM linear model, with kernel = ‘linear’, class weight ‘balanced’, verbose = True and C = 1. The model was more successful in predicting the throws missed but overall had worse performance.

When we tried the Neural-Net model with binary cross entropy loss we got really bad results on the shot missed, the model had predicted shot made for all the throws.

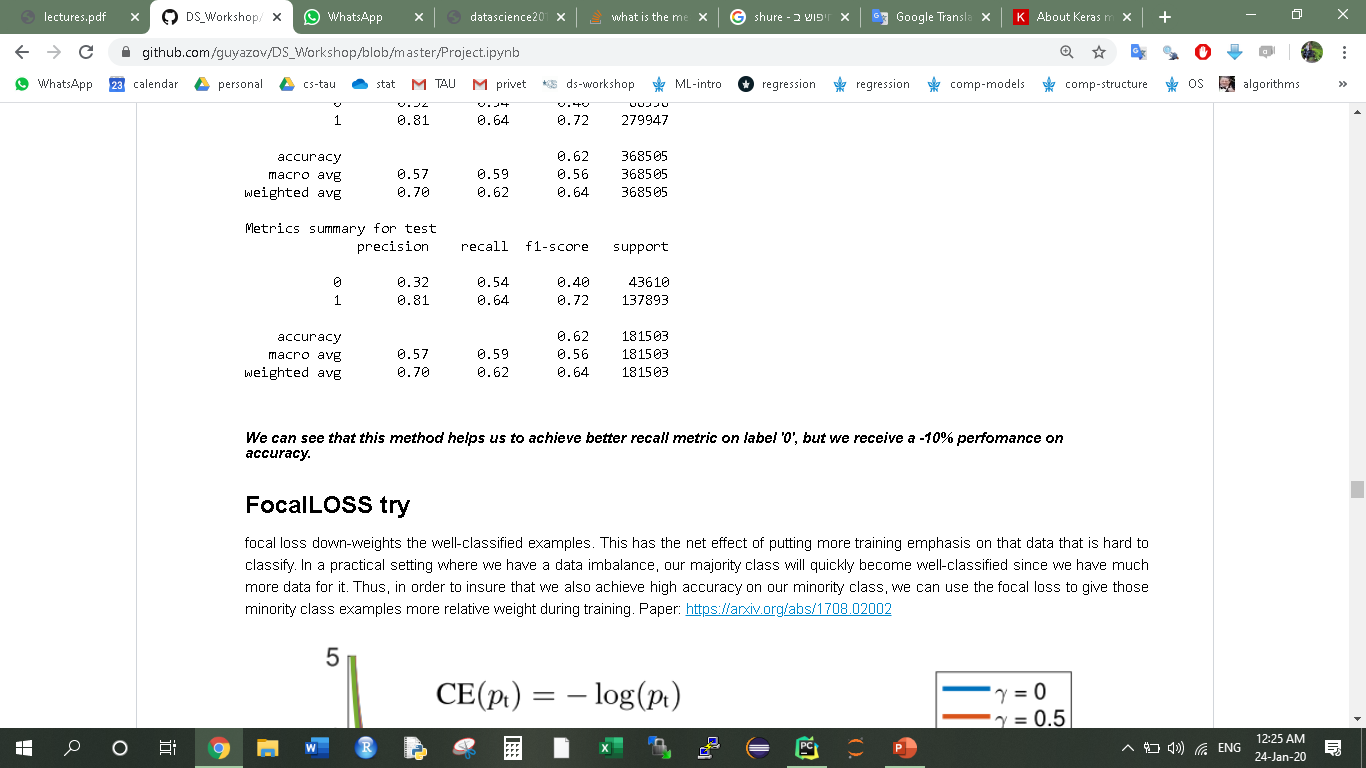
And also in the Xgboost regressor model and Xgboost classifier model

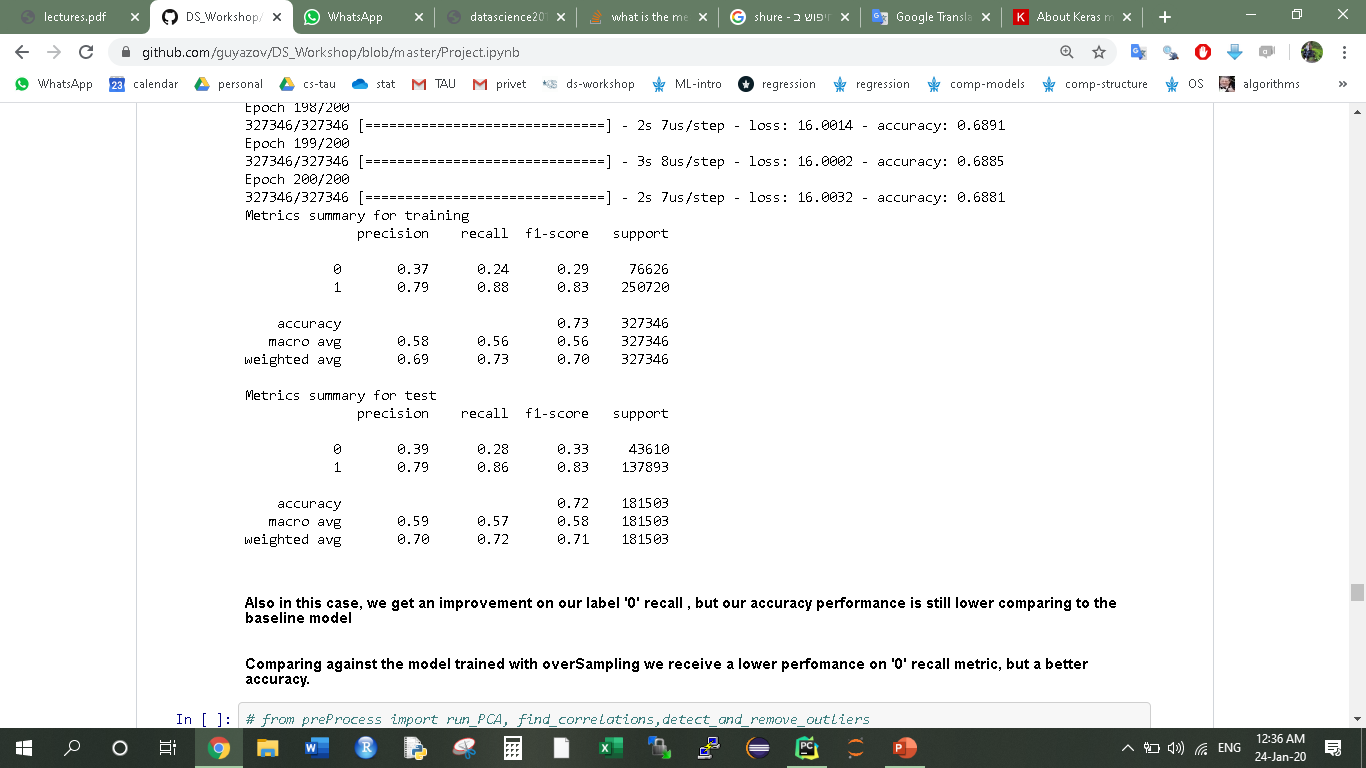
We ran a random forest model with the parameters Estimators = 16 and max. tree depth = 20 and saw a slightly improve

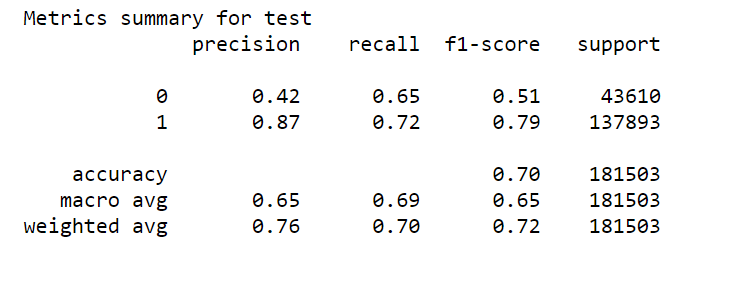
And in K-Nearset Neighbours Model with the parameters K = 5 and metric distance – Eucledian. Our recall got better but our accuracy was smaller, NO FREE LUNCH.

**Phases VII+VIII: Evaluate your model statistically and use your results to improve the feature selection and parameter tuning, Present the important and interesting parts of your workflow and results in a user friendly human-readable Jupyter Notebook**

We understood that because our dataset is un-balanced, the models tend to predict '1' (shot made) most of the time, as we saw for instance by the confusion matrix of the logistic regression model. This tendency to predict mostly ‘1’ cause the recall of the lower represented class (‘0’) to be extremely low. Our goal is to make the recall higher while not decreasing the accuracy metric. So, we tried to use methods for imbalanced data:

OVERSAMPLING:We oversample the smaller class, so there is an equal number of misses and scores in both train and test. This can be achieved by simply duplicating examples from the minority class in the training dataset prior to fitting a model. We used the SMOTE (Synthetic Minority Oversampling Technique) which works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. We checked this oversampling method with a NN network and saw that this method helped us achieving better recall metric on the ‘0’ class, but we received a -10% performance on accuracy.

FOCAL-LOSS: Another way to confront the unbalanced dataset is the Focal loss. It down-weights the well-classified examples, so we used it to ensure that we also achieve high accuracy on our minority class by giving them more relative weight during training. We got an improvement on our label '0' recall, but our accuracy performance is still lower than the baseline model.

Increasing the decision threshold: Another strategy is to make the model choose ‘1’ only if it has “confidence” with this choice (determined by a pre-defined threshold). Instead of classifying a datapoint to ‘1’ category if , we decide to search for a more confident threshold for this. For example, after grid-searching the best achieved recall by the threshold parameter, we got with a Random Forest model, and we saw that the recall was a lot higher while we didn’t heart the accuracy badly.

**Phase IX: Derive interesting insights and/or applications from your analysis**

As we explored our models and tried to understand their behavior, we noticed that we have very similar datapoints belongs to opposite classes.