LAB 8 - Random Forests

In this lab session, we are going to implement random forests with many different parameters, visualize the results and try to examine the differences between different parameters.

We are going to use sklearn's `RandomForestRegressor` class for this task. You can examine its documentation here:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Random forests have many parameters which we can alter to control tree characteristics. We are going to investigate 3 of those in this lab: `max_depth`, `n_estimators` and `max_features`. Please browse through the documentation link given above to clearly understand what each of those parameters refer to.

- (10 pts) Build the input and output:

Our .csv file is the same one we used for the previous lab. Our input (X) consists of 7 columns. Combine the following into a matrix (Remember, they have to be *columns*, not *rows*):

FSP.1 First Serve Percentage for Player 1
ACE.1 Aces Won by Player 1
DBF.1 Double Faults Committed by Player 1
WNR.1 Winners Earned by Player 1
UFE.1 Unforced Errors Committed by Player 1
BPC.1 Break Points Created by Player 1
NPA.1 Net Points Attempted by Player 1

Our output is going to be the total number of sets won by player 1. This corresponds to **the *sum* of 5 different columns as a 1D array** in the .csv file:

ST1.1 Set 1 Result for Player 1
ST2.1 Set 2 Result for Player 1
ST3.1 Set 3 Result for Player 1
ST4.1 Set 4 Result for Player 1
ST5.1 Set 5 Result for Player 1

Simply sum each of these columns to form your output vector (y).

Divide your input and output into two parts: First 200 entries are going to be training data, and the rest will become test data.

- (50 pts) After building your X and y for train and test data, do the regression fitting in a loop from `n_estimators` = 1 to 150.

  Inside this loop, you should calculate three different regressions for:
    - `max_features = "auto"`,
    - `max_features = "sqrt"`,
    - `max_features = 4`.

  For each of these fitting processes, your `max_depth` parameter should be 7. An example of regression fitting is as follows:

  Simply replace these parameters with corresponding values inside the loop to create the regression object:

```
reg = RandomForestRegressor(max_depth=1, n_estimators=150, max_features=4)
```

  And then call the `fit()` function of the regression object:

```
reg.fit(X_train, y_train)
```

  Where `X_train` is your training input, and `y_train` is your training output.

  After fitting, you should get predictions using the test data:

```
predicted_y = reg.predict(X_test)
```

  While doing the fitting, you should also keep 3 different arrays (of size 150) for R^2 scores, one for each regression, for later plotting purposes.

  After getting the predictions, compute the R^2 scores. Put these values into the corresponding arrays.

- (10 pts) After the loop ends, do the regression two more times for:
  `max_depth = 7`, `n_estimators = 150`, `max_features = 4`, and
  `max_depth = 1`, `n_estimators = 150`, `max_features = 4`.

  Then, do predictions using your test data for both regressions. Save these two sets of predictions as "`y_pred_1`" and "`y_pred_2`".

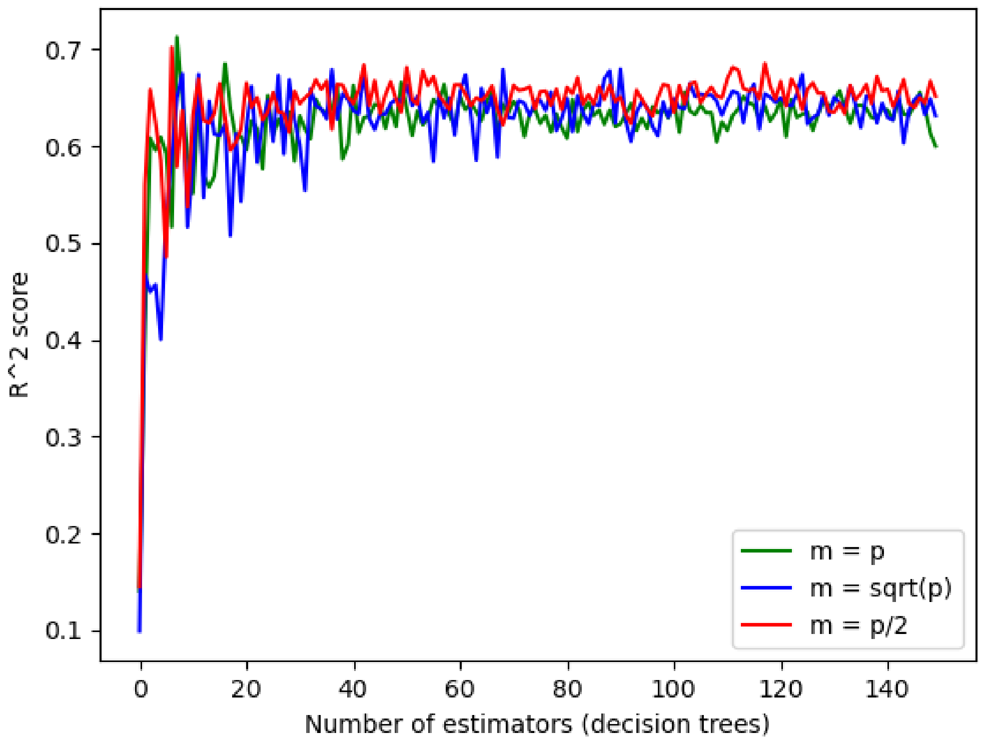
- (30 pts) Now, the plotting. You have two different plots to make, and these plots should be **on separate figures!**

  - ❖ For the first figure, you will use the R^2 score arrays. These will show how error rates behave for different parameters.

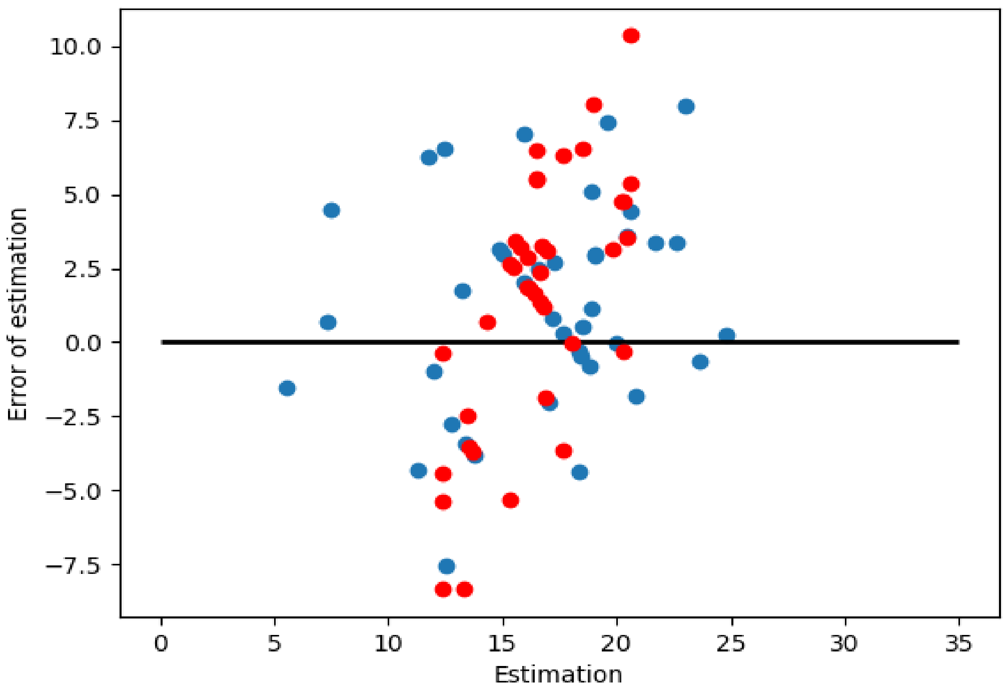
    Call the `plot()` function three times. For each plot, your x-axis will be simple arrays consisting of integers going from 1 to 150 (you can use the "`arange()`" function of `numpy` for this as we did in the previous lab), and your y-axis values will be the R^2 score arrays. Please plot each line in a different color. It's also useful if you implement a legend.

  - ❖ For the second figure, you will create two estimation-error plots. Both plots will be on the same figure as scatter plots, with different colors. It's very important to be able to interpret these plots. Simply call the `scatter()` function twice:
    - ➢ Once using "`y_pred_1`" as your x-axis, and the error of those predictions as your y-axis.
    - ➢ Once doing the same with "`y_pred_2`".

Here are example figures:



R^2 score
Number of estimators (decision trees)
m = p
m = sqrt(p)
m = p/2



Error of estimation
Estimation

Continuing with the insight section:

Understanding how random forests work is crucial for interpreting what we did in this lab. The first figure we generated depicts the behavior of “bagging”. The parameter “`n_estimators`” here shows how many copies of the data we should create. These copies are random subsets of our training input, which are used to build different decision trees. These trees are later combined to create a single decision tree.

Bagging is a very useful approach to generalize a model, but only up to a point. If we use it too much, we’re simply creating similar datasets repeatedly, and creating similar decision trees repeatedly, which doesn’t affect anything. It’s only extra computation and overhead without improving the error rate.

The importance of the depth of the tree is shown in the second figure. You can see that when the depth is too low, we get less varied predictions with a higher error rate. But you should also be careful not to increase it too much due to the risk of overfitting!

Please feel free to tinker with the parameters and observe the effects. It’s one of the best ways to learn!