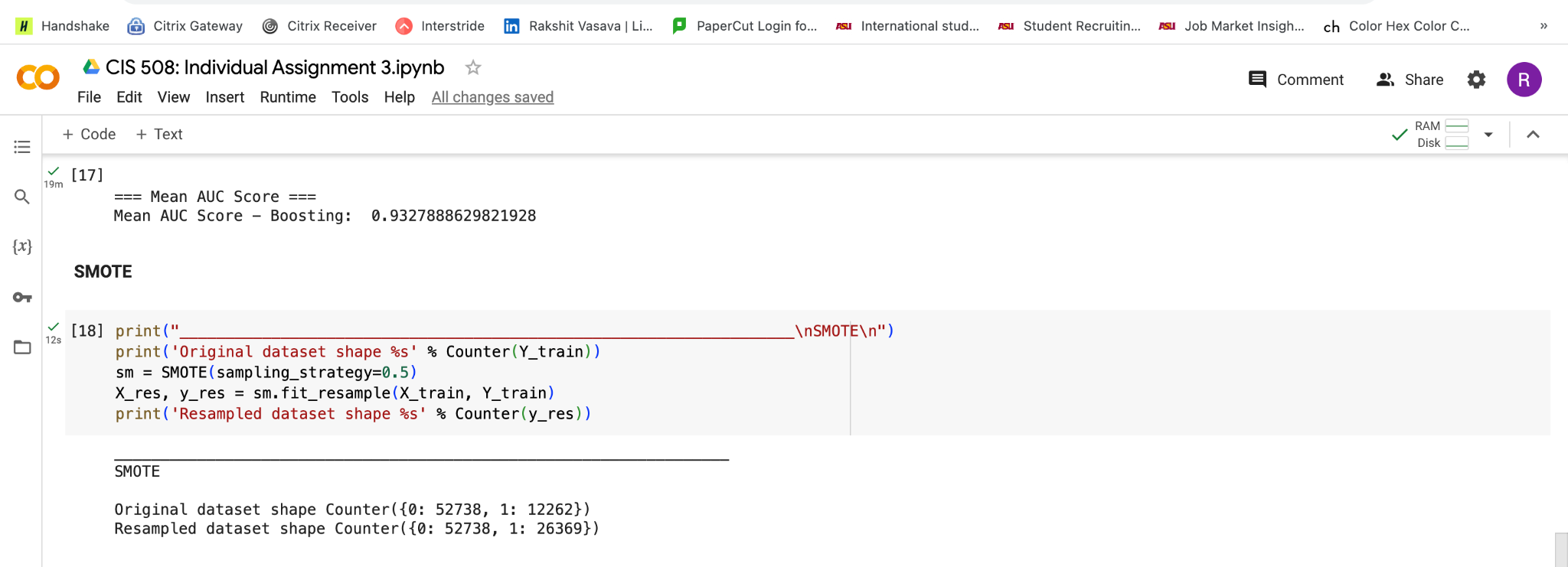
**Homesite Quote Conversion Problem**

**Rakshit Vasava**

We have to do a Kaggle submission for the homesite quote conversion problem. We are given train data and test data. We also need to use the stacking model to perform the prediction. The prediction will be whether the customer will buy the insurance or not.

**To-Do List and Code Explanation:**

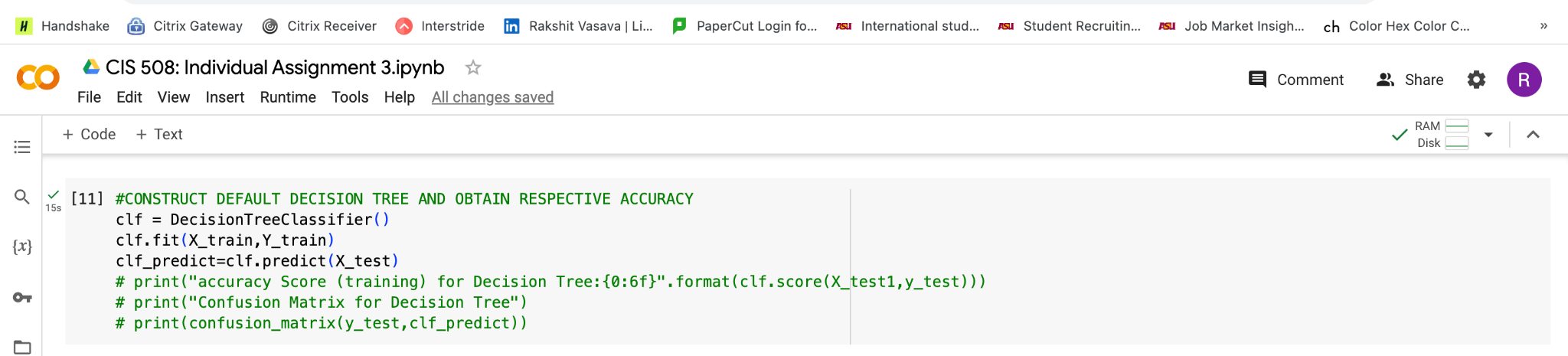
***1. Experiment with SMOTE (or its variations) using different percentages to get a higher accuracy on minority class prediction.***

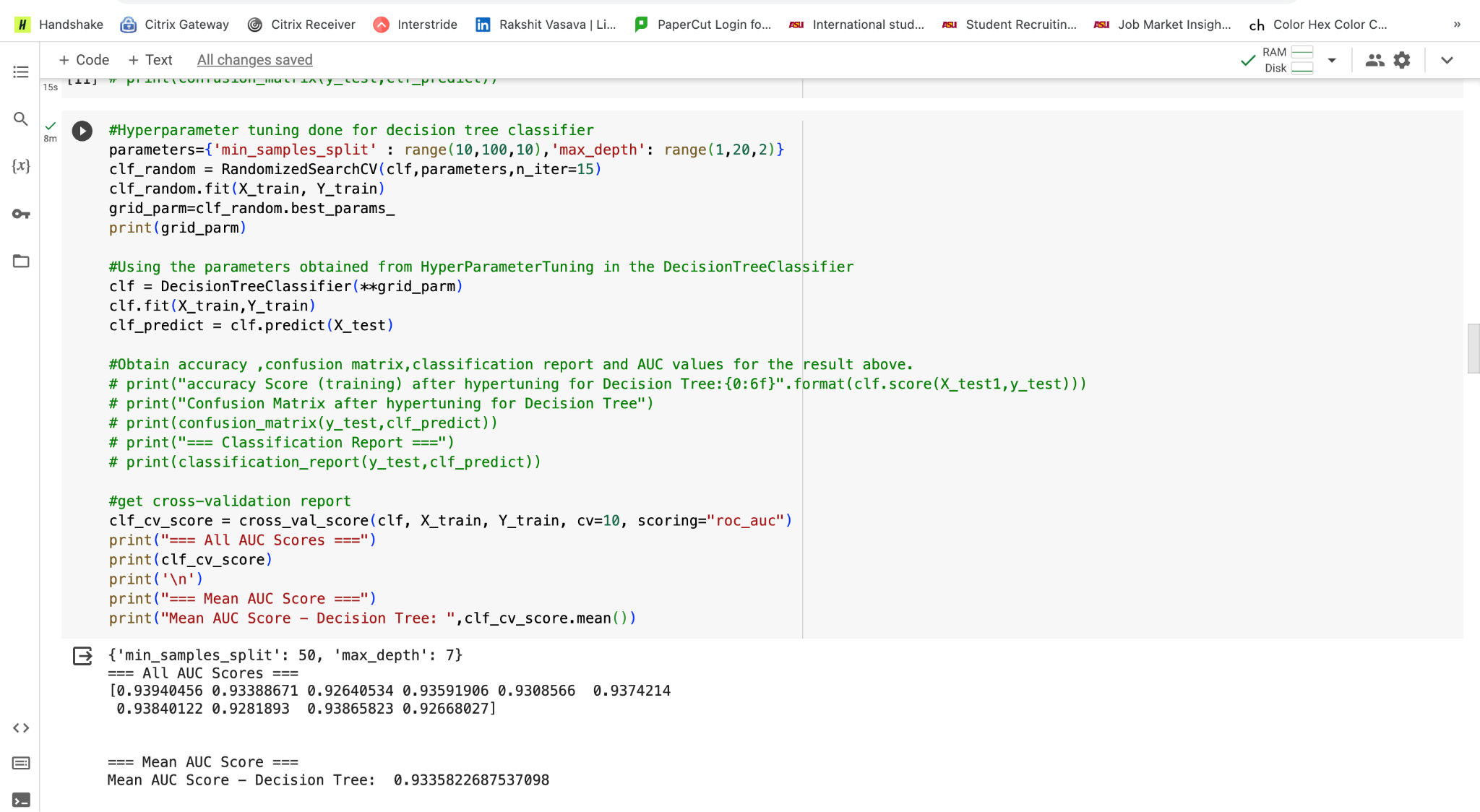


In the code above I’ve used the SMOTE functionality to generate new synthetic minority class values to deal with the imbalance dataset problem. Here, I’ve taken the ratio as 0.5. So, the minority class will be 0.5 of that of the majority class.

***2. Perform ensemble predictions (one-layer stacking) by combining predictions from the various algorithms. For stacking, try at least five different models - e.g. decision tree, random forest, support vector machines, multilayer perceptron and K-nearest neighbors.***

I initially did individual model building for the different classifiers, namely: Decision tree,

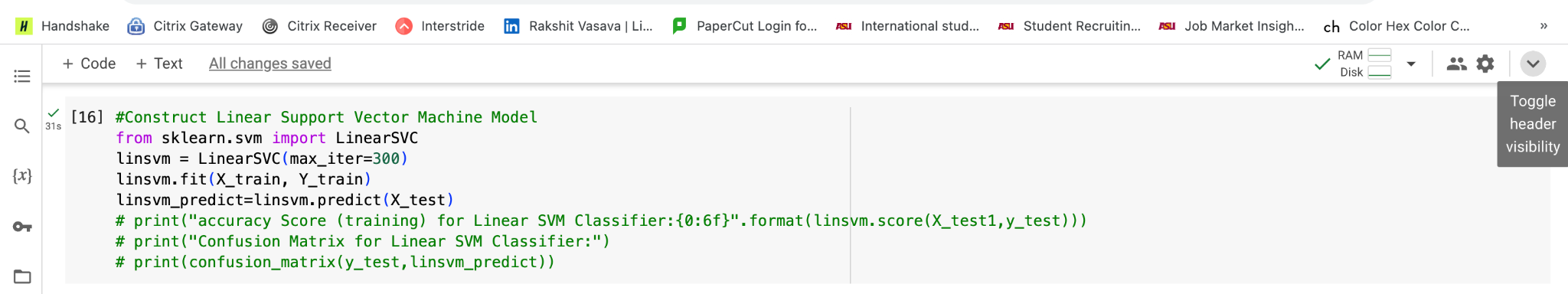




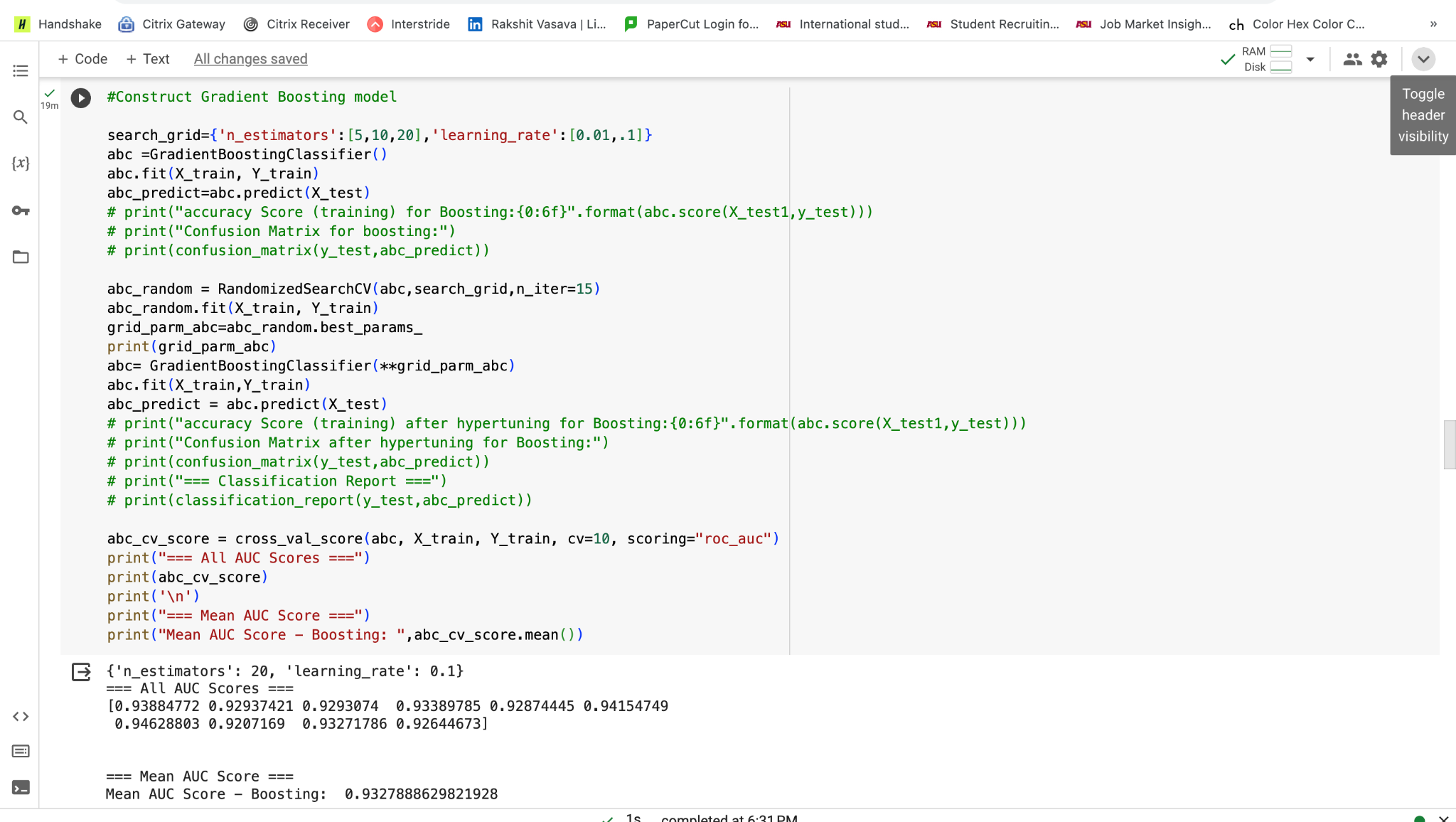
Random forest, Multilayer perceptron Model, K-nearest neighbor model,



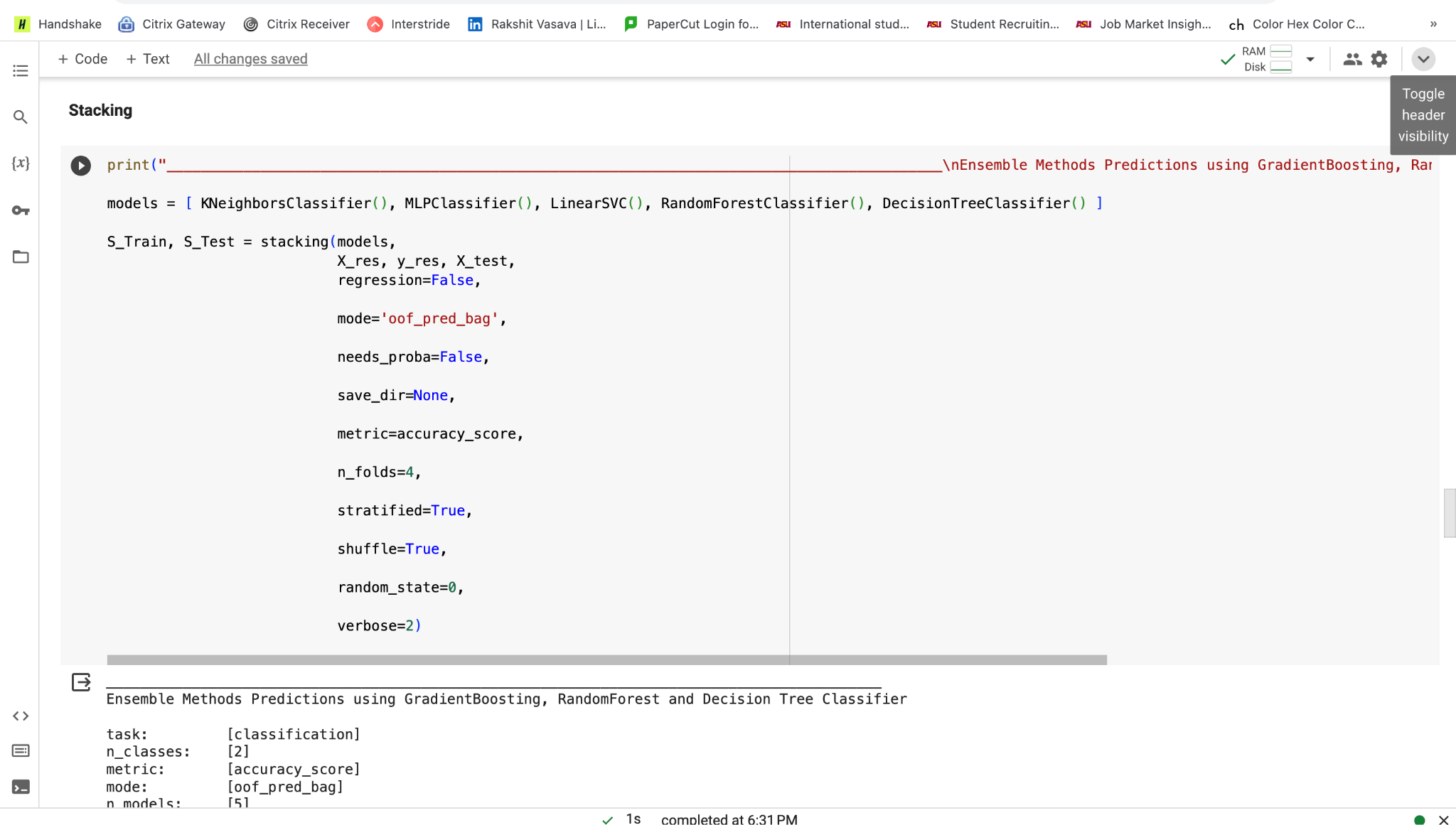
Linear Support Vector Machine,

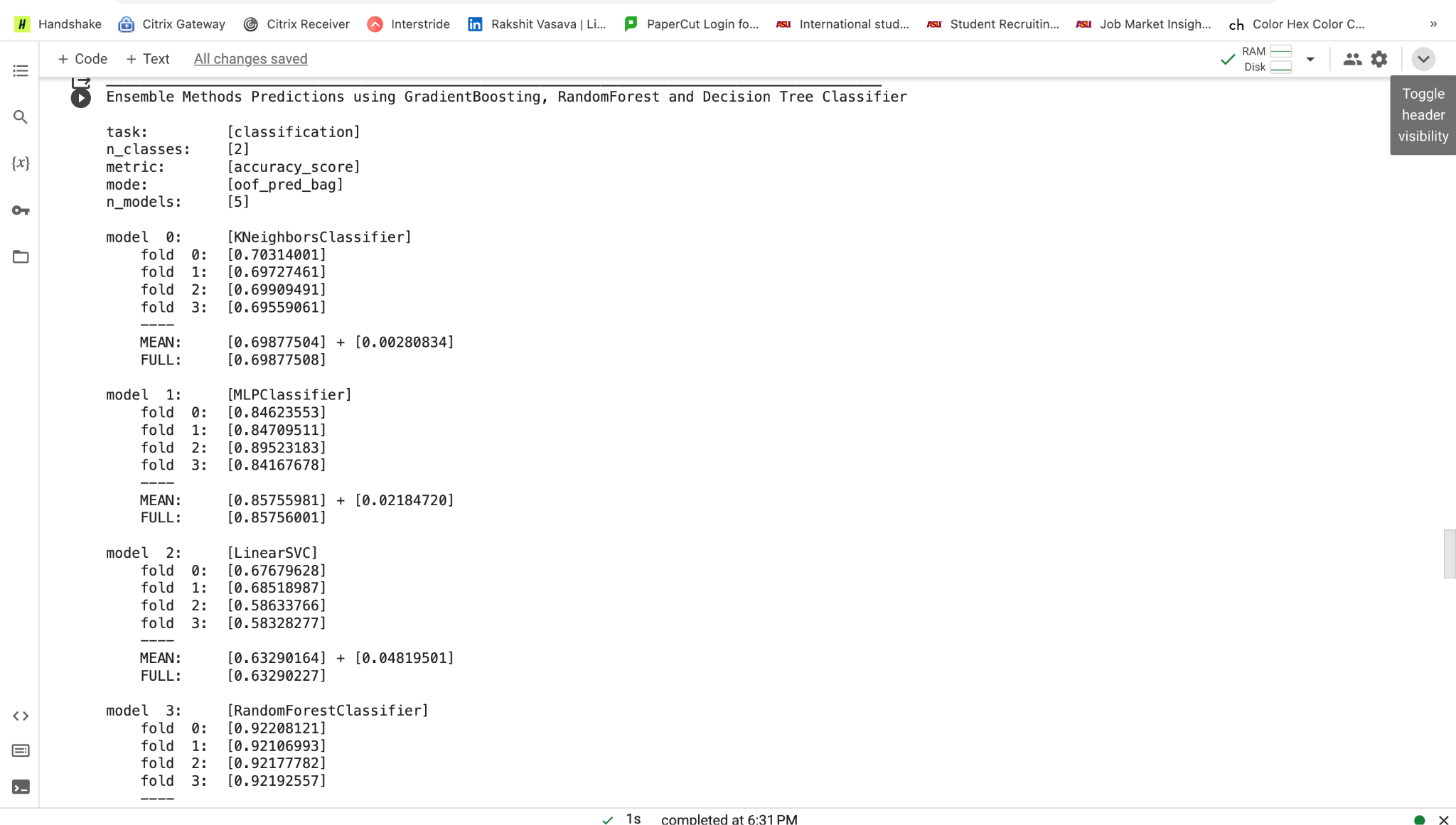


And even did the Gradient boosting model.



And then after performing SMOTE, I did stacking taking the 5 base models as K-nearest neighbor, Multilayer perceptron, Linear Support Vector Machine, Random Forest, and Decision Tree.



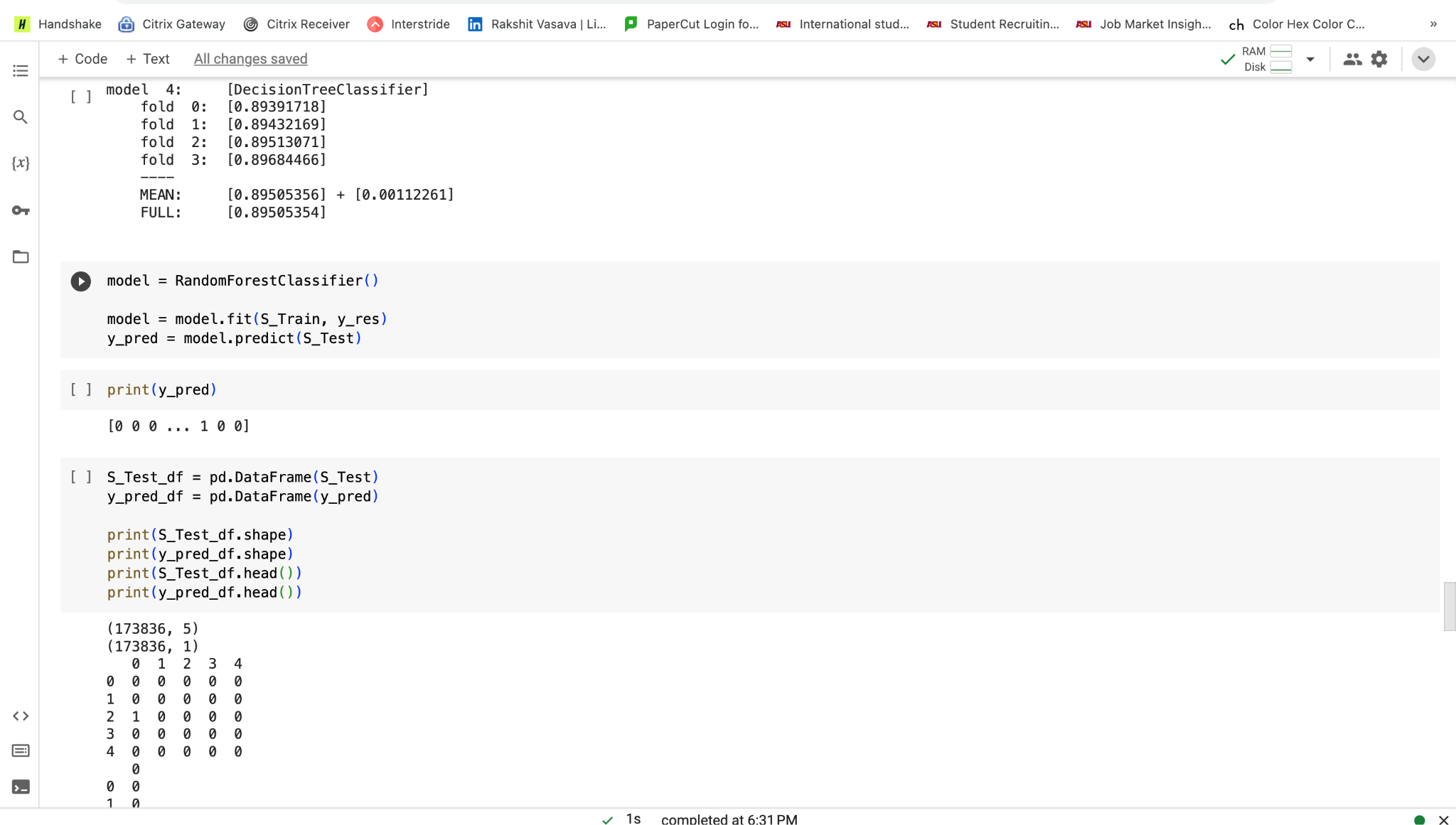


Once the stacking model is built, we get two separate data sets, S\_Train and S\_Test.

S\_Train - Stacked predictions (meta-features) on the training data obtained from the base models.

S\_Test - Stacked predictions on the test data obtained from the base models.

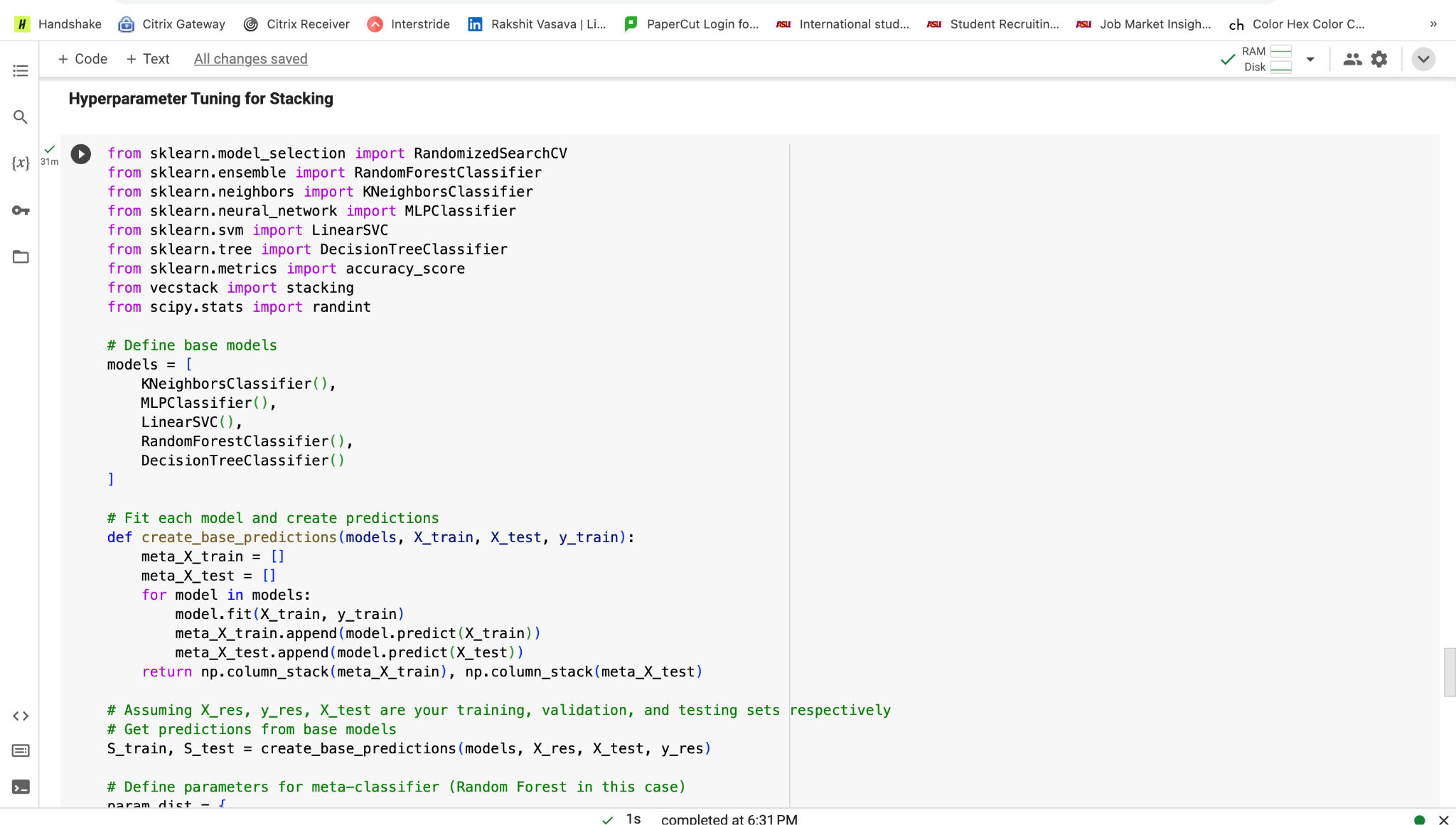
These are then used to train a different model say, a Random forest classifier.

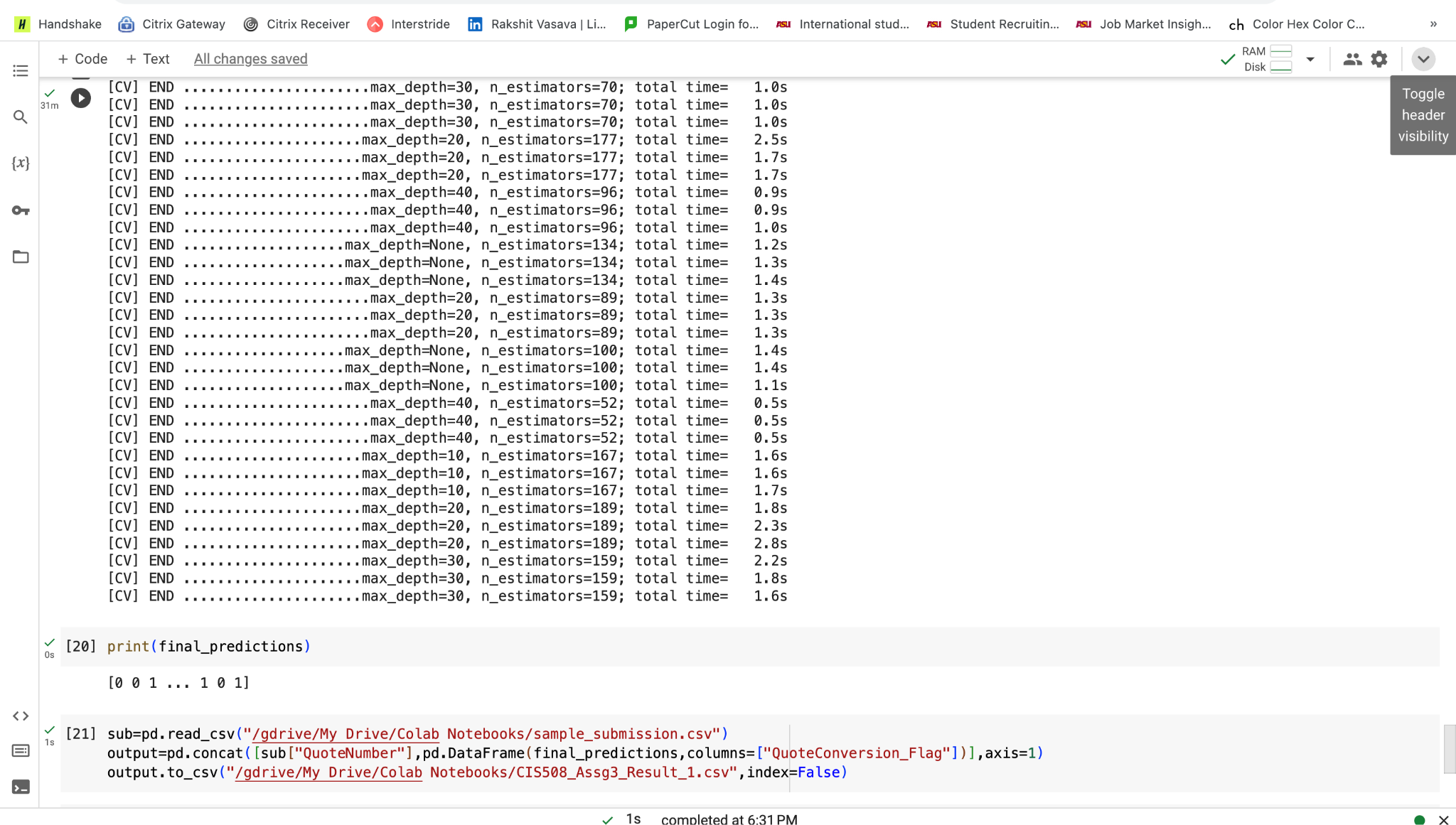


The results obtained from this are then converted to a CSV file and submitted on Kaggle.

***3. In addition, perform hyperparameter tuning on the stacked model. You can do hyperparameter tuning on individual models if you want, but that is not necessary.***

I have done hyperparameter tuning for the stacked model and tried to submit its score and it did improve.

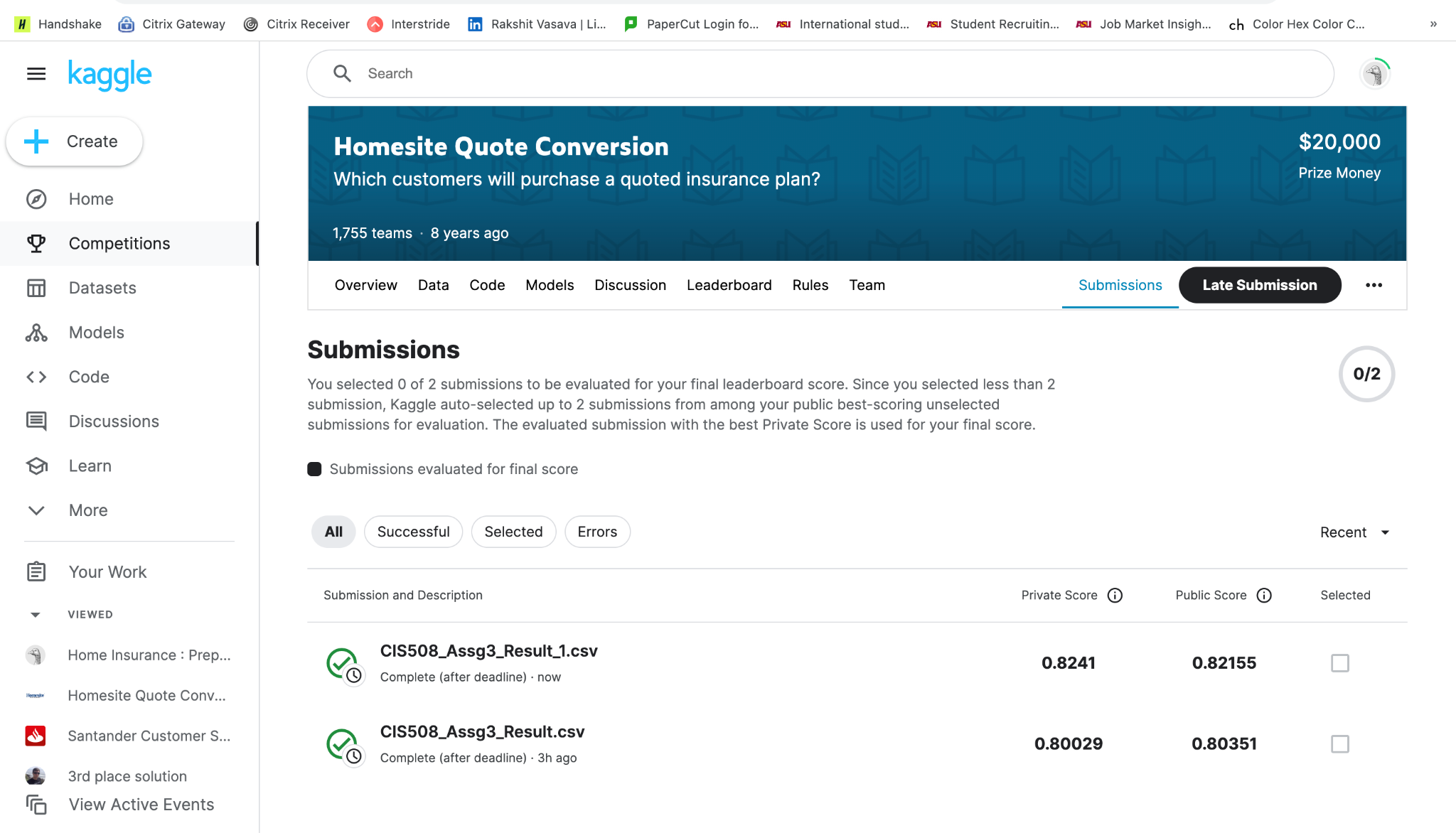




For this also I have used the random forest as the meta-classifier.

***4. Submit to Kaggle the stacked model predictions and report all such Kaggle scores in an Excel table.***

I did two submissions on Kaggle and got the following result.



For the normal stacked model with SMOTE used values, I got a score of **0.80029**(private), **0.80351**(public), and for the hyperparameter tuning of the stacked model, I got a score of **0.8241**(private), **0.82155**(public).