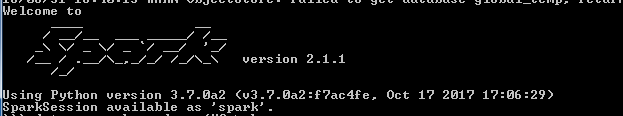
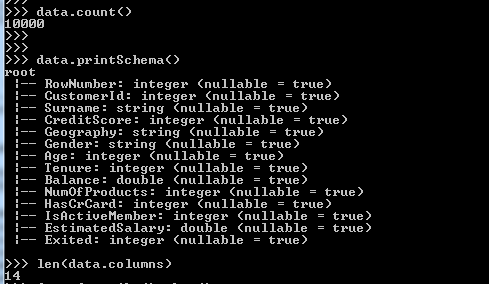
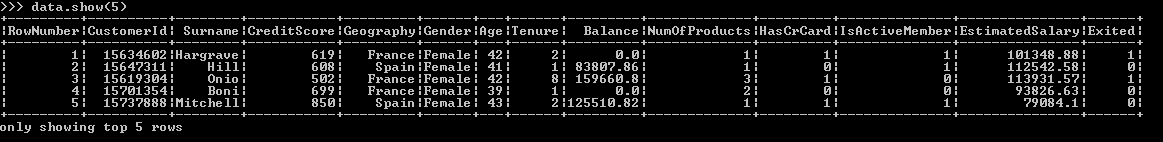
I will perform exploratory analysis on the Bank Churn Modelling dataset, where the scenario is that clients are leaving a bank. The dataset is the same as in a previous project, but I will be working in an apache environment, using pyspark and sparksql commands.



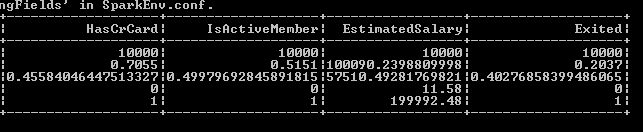
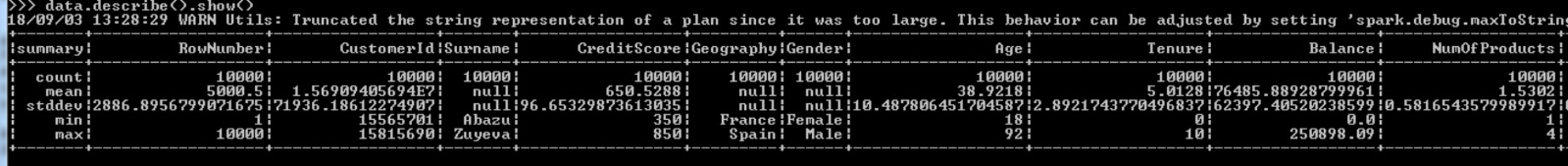
Loading the csv file from my C:\ driveC:\Users\tolaros\Desktop\pyspark project\loadingcsv.PNG

Some basic exploration





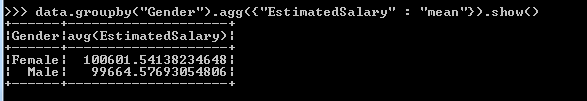
Our data consists of 10000 rows and 14 columns. But first let us run a general command.



We can see that for the gender and geography features, there is no result for mean and stddev, because they are categorical. Another remarkis that, there are 5457 males and 4543 females in our dataset.

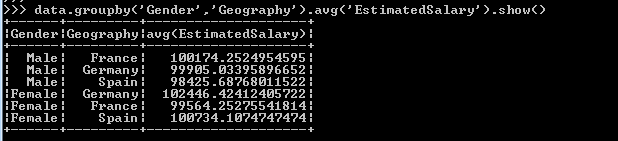
Looking at the 'Exited ' column', which is our label, 20% of the customers, are leaving the bank. Our dataset's ratio is 80-20, not a great, but still a significant imbalance.

Moving on with our exploration, I would like to see the avg salary per gender in this dataset.



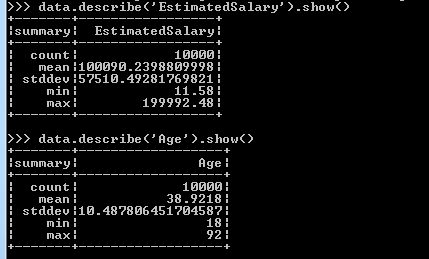
Females, slightly have a better salary.

Let us now see the above, but including the country as a parameter, and with a ‘tweak’ in the syntax:

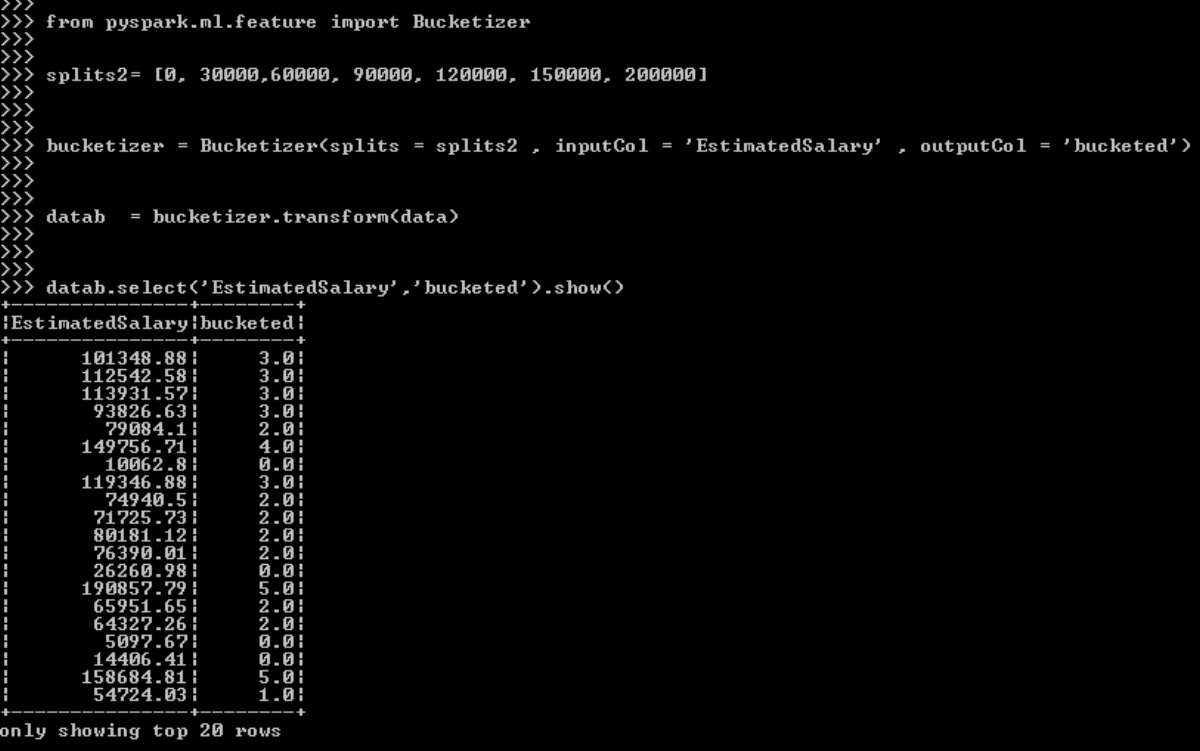


**BUCKETIZATION**

Another useful manipulation could be Bucketizing some features.



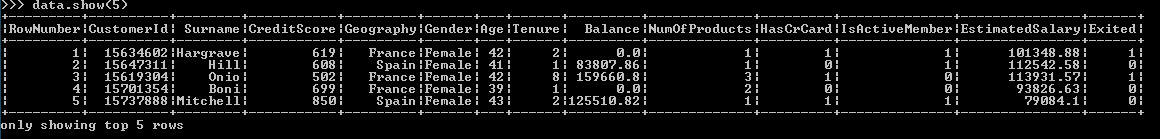
Salary is between 0 - 200000, and Age is between 18 -92. I will bucketize the salary column, creating 6 buckets, which are declared using the splits2 variable.



The ‘bucketed’ row, shows exactly to which bucket each value corresponds

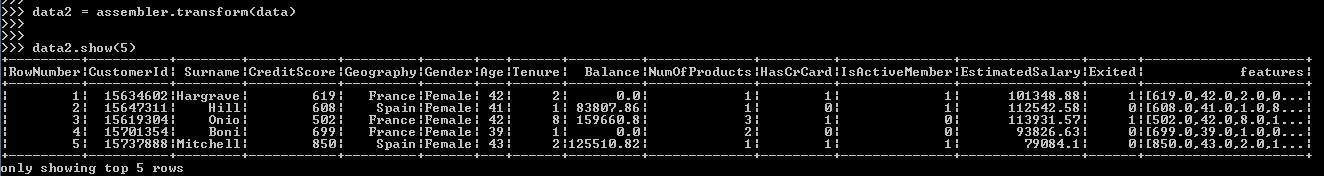
**BUILDING A MODEL**

Now let us apply a model to predict if an individual will leave the bank. I want to take again a look at the data



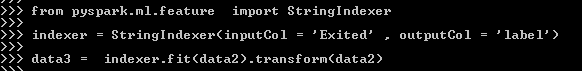
I will not use CustomerId or Surname. They do not have an impact on our result. I will use only those features, that are already numeric. So let us remember exactly who they are, as I place them into the assembler object:C:\Users\tolaros\Desktop\pyspark project\assembler.PNG

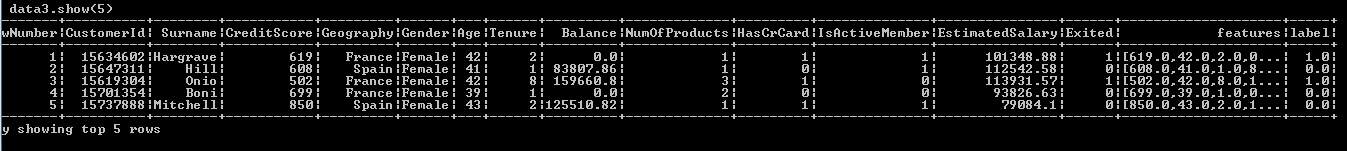
C:\Users\tolaros\Desktop\pyspark project\assembler2.PNG



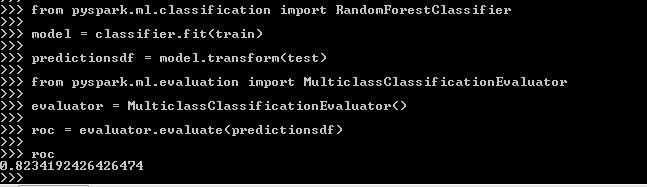
As we can see, a new column called ‘features’ was added

Next on, we should label our target variable, ‘Exited’.





We have created an extra column, ‘label’.

Next on we will try a RandomForestClassifier and we will evaluate it also

We obtained a rocauc score of 0.823, which is quite decent.

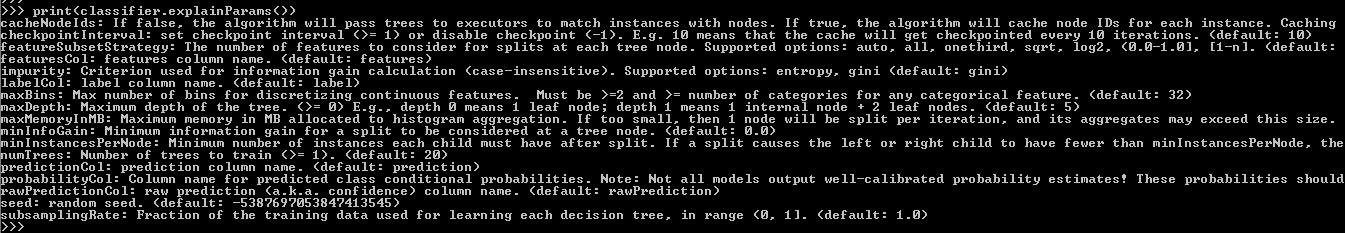
Our classifier has a built in feature importance attribute. Let us see, which are the most importantC:\Users\tolaros\Desktop\pyspark project\randomforestfeatureimportances.PNG

So, features 1, 4 and 6, are in descending order the top3 of importance. Below, I am refreshing our memory on the

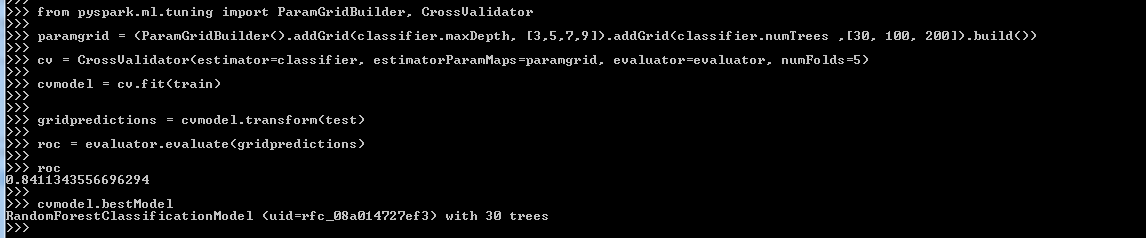
sequence of the features.

C:\Users\tolaros\Desktop\pyspark project\ASSEMBLER\assembler.PNG

1 = ‘Age’ , 4 = ‘NumOfProducts’ , 6 = ‘IsActiveMember’

Next on we will evaluate our classifier through hyperparameter tuning. But first, let us take a look at the parameters

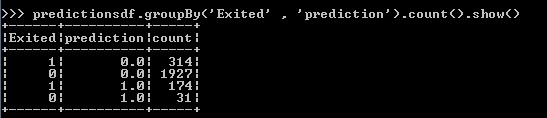
I choose to tune two very common parameters, numoftrees and maxdepth.



Some remarks:

1. Our roc auc improved to 0.841.
2. The best num of trees is 30

Now, let us take a look at the confusion matrix. I will give a sort of SQL command to have a better understanding of the predictions.



Manually now:

Accuracy = (1927+174) / (314+1927+174+31) = 85.9%

Precision = 174 / (174+31) = 84.8%

So, our model , besides having a very good roc auc score of 84% , also has a very good accuracy and precision.