**Git Repository Link**

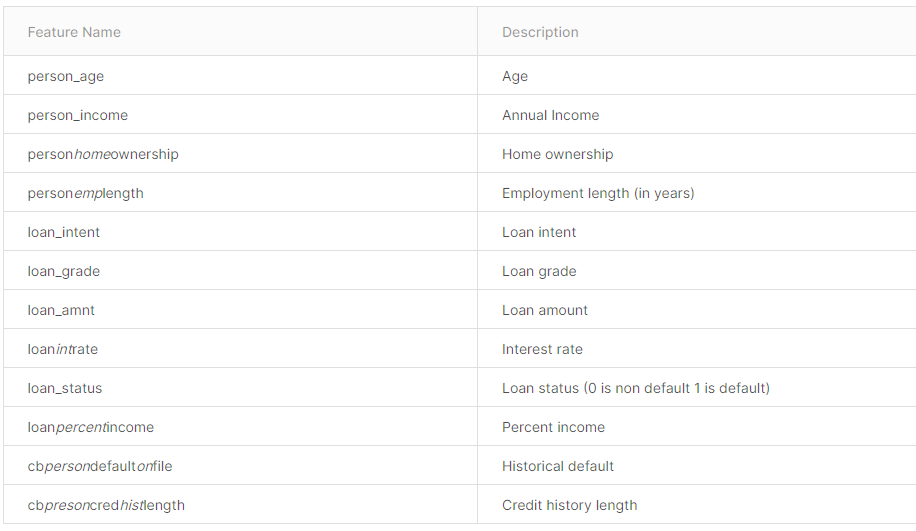
[**https://github.com/vbabar/Credit\_Risk\_Supervised\_Learning**](https://github.com/vbabar/Credit_Risk_Supervised_Learning)

**Project Goal**

Project goal is to predict if customer will default on a loan. This is help bank to minimize the risk so that they can decide on whether to provide a loan to the prospective customers.

**Dataset :** I used the dataset available on a Kaggle

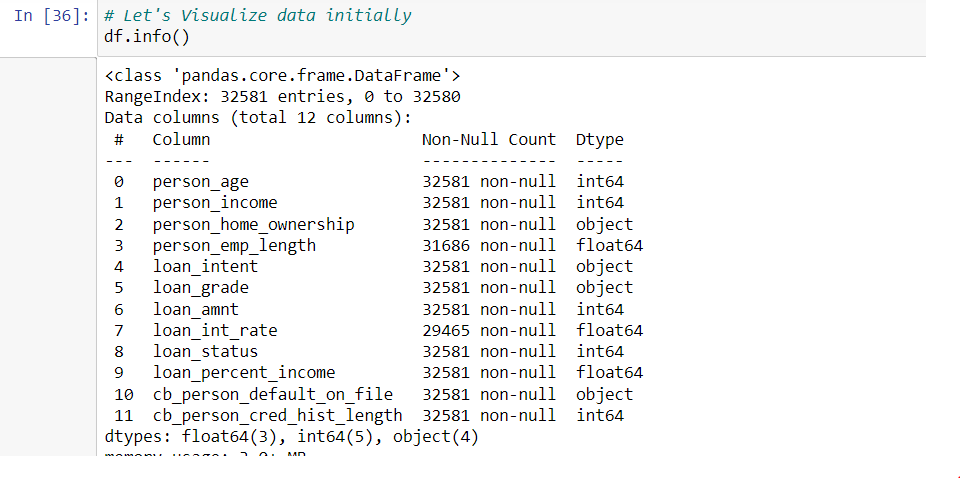
**Dataset Columns and Description** :- Dataset has following columns

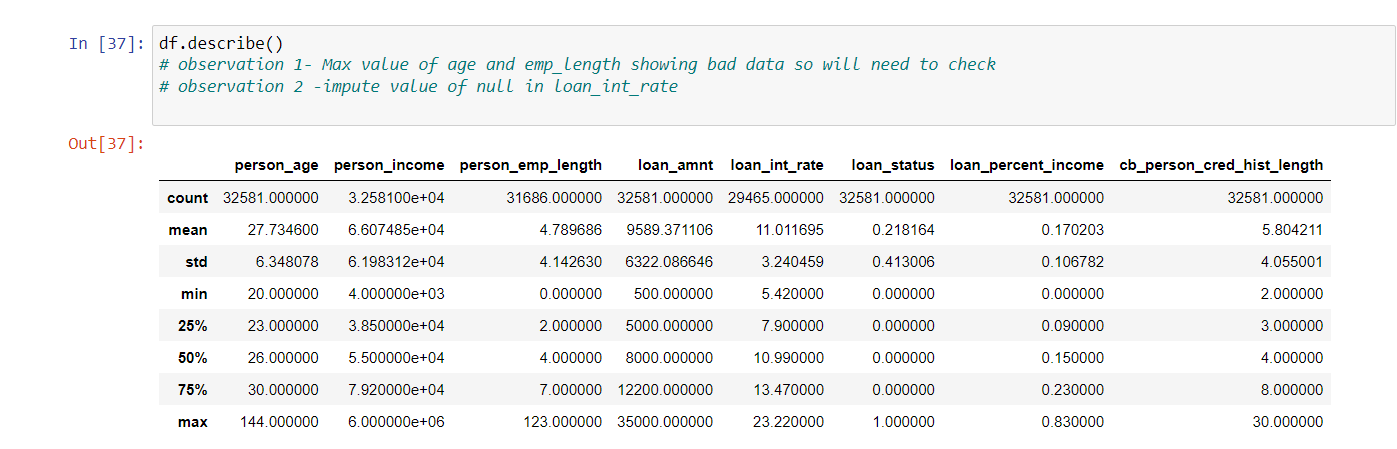


**EDA**

1. Let’s visualize the data initially

* Shows that loan\_int\_rate has null value
* Check on percentage of null and histograph before determining the imputing method.
* Third – think of one-hot-encoding for ‘discrete’ variables – not sure if logistic regression will work with them with ‘discrete value’ –
* Try – model on two dataset – one with ‘discrete value’ and other dataset without -discrete value.





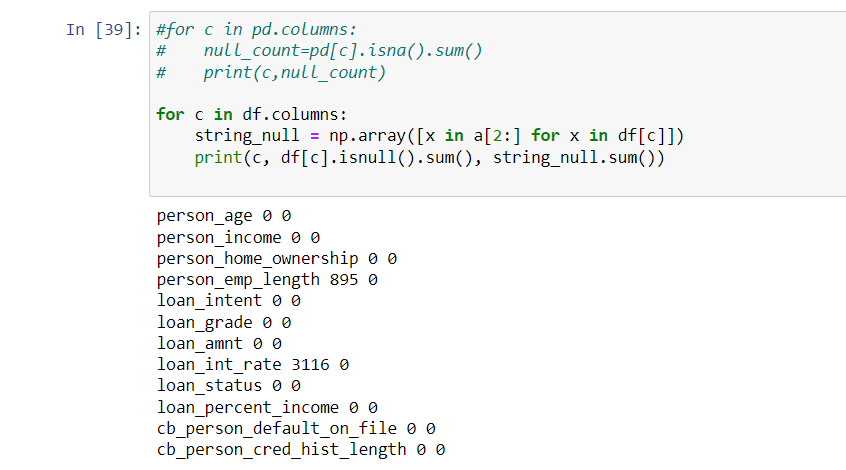
Statistical Findings:-

1. Total count of records – 32581
2. Obvious finding – loan\_int\_rate has null
3. Persona\_age and person\_emp\_length – inspect them as max value is 144 for person age and person emp length is 123 – so check out for quality of data too.
4. So first decide on “impute” and then go for ‘Box plot’ based on ‘whether he will default or not’
5. Then go with one-hot-encoding

EDA 1 :- Bad data finding –

Two columns -has bad data –

1. Person\_emp\_length
2. Loan\_int\_rate



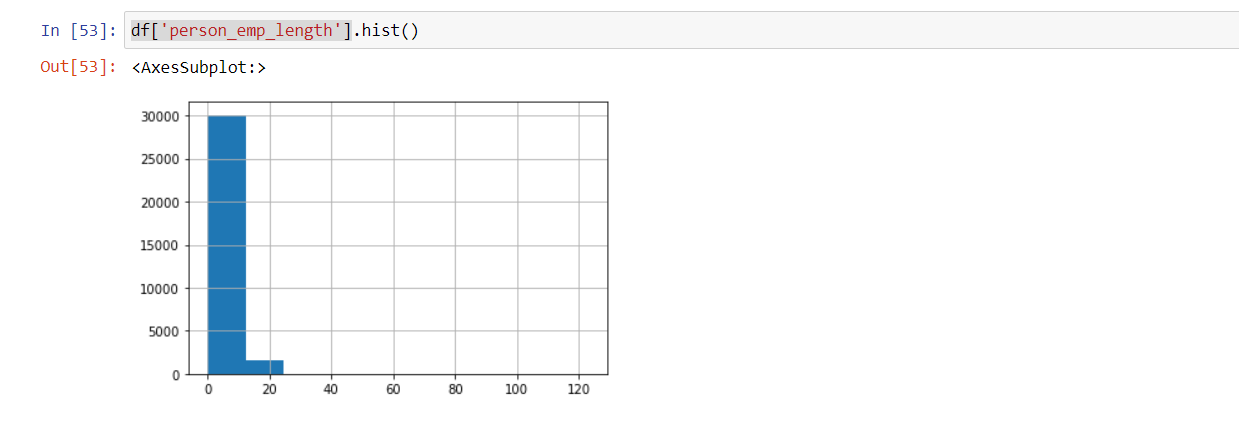
1. **Decided to drop a ‘column’ – “Loan\_Int\_Rate”** as 9.56% of records has ‘bad value’ in this column.

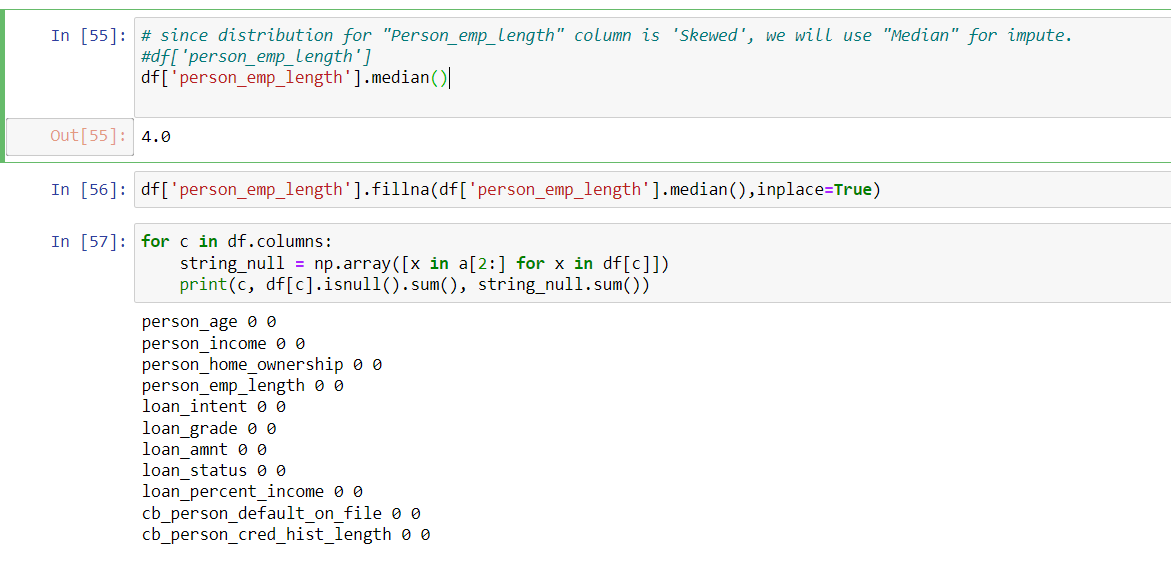
We would have looked in deep had we met with SME.



1. Let’s see the distribution for “person\_emp\_lenght” as it too has null value.

* Since data is skewed, we decided to use the “median” method for impute.



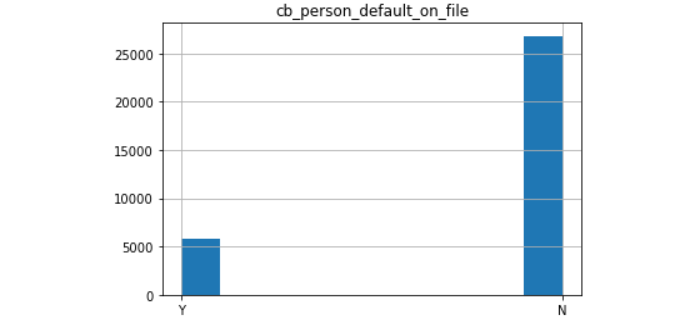


**Let’s understand the data for each column**

**Unbalanced Data**



* One interesting observation – data is not balanced ..most of observations have not defaulted.
* So we need to remember that we choose the ‘parameters’ for the model.

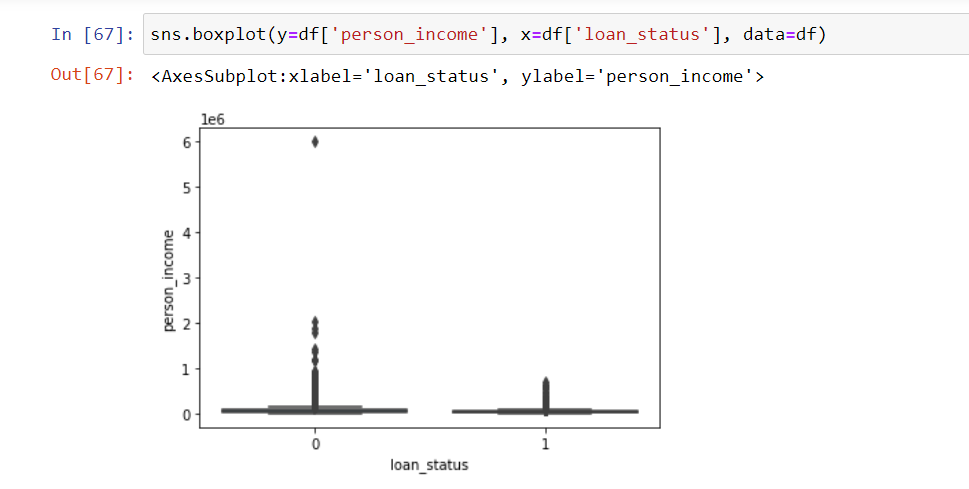


Let’s do the “Outlier Detection”

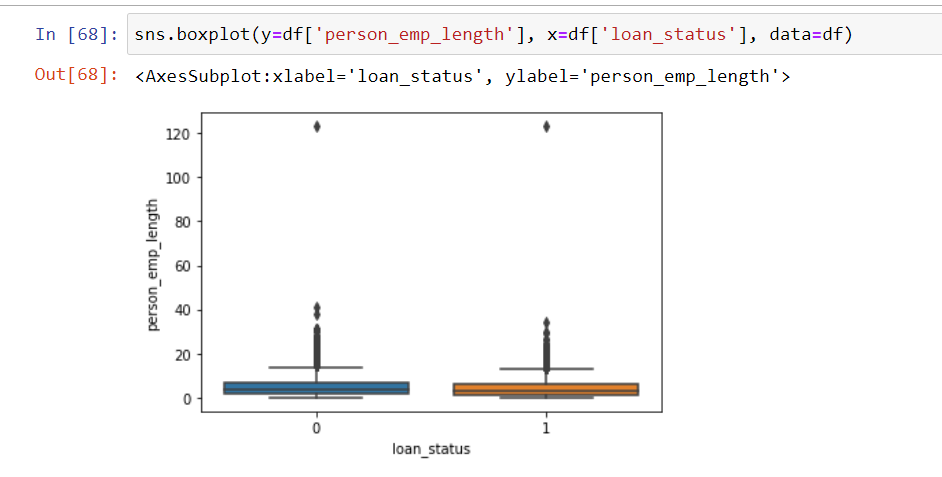
* You can use “Box Plot” for visual inspection
* Let’s start one by one – we see – outlier – in person age
* Let’s see what you see ‘using interquartile range’



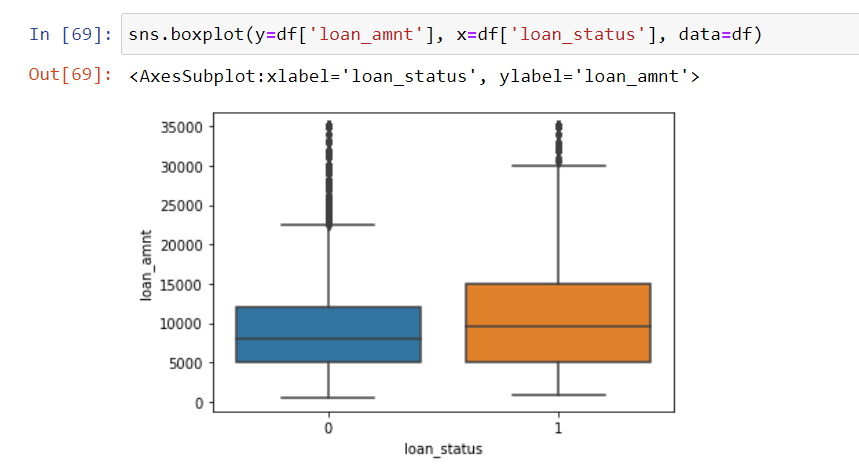
We also see outlier in a “person income’ too.



Outlier for person emp length too



Observation :- The one who are defaulted ( Loan\_status=1) – has higher median loan..so they likely have take more loan compared to one who have not defaulted.

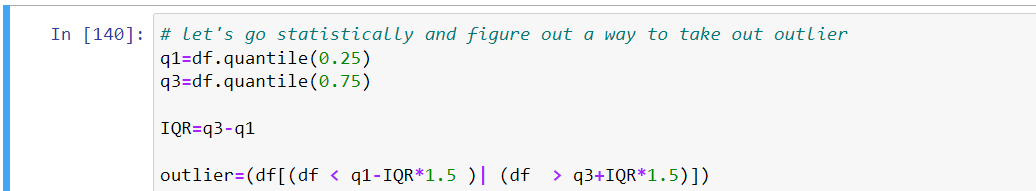


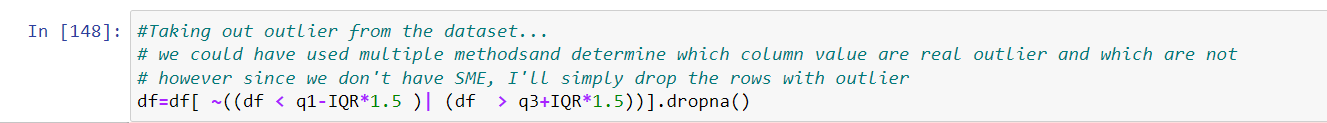
Similar status we see using following variable –“loan\_percentage\_income”, the one who has defaulted has higher ‘loan percentage to income’ ratio

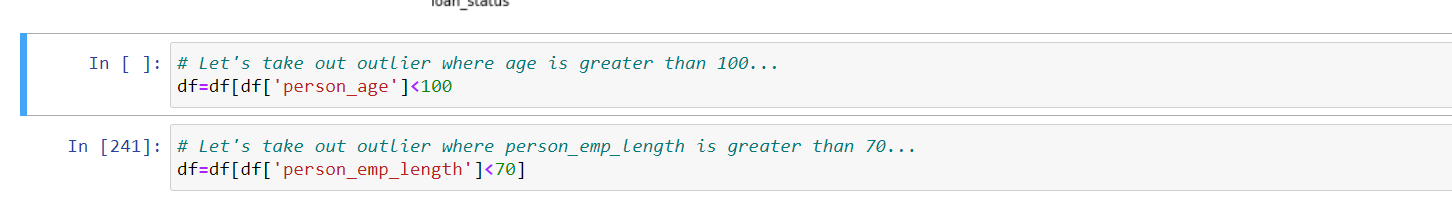
**Very good reference for outlier**

[**https://careerfoundry.com/en/blog/data-analytics/how-to-find-outliers/**](https://careerfoundry.com/en/blog/data-analytics/how-to-find-outliers/)

**Let’s take outlier using IQR formula**



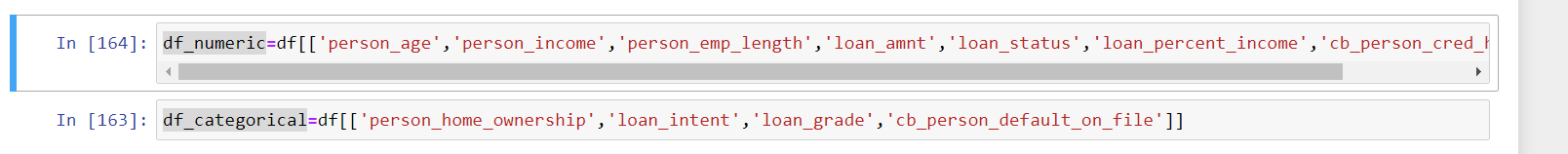


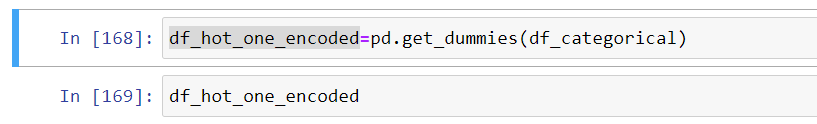


**Feature Engineering**

1. **One hot encoding for categorical variables**

**Let’s do the one hot encoding for the ‘categorical variables’**

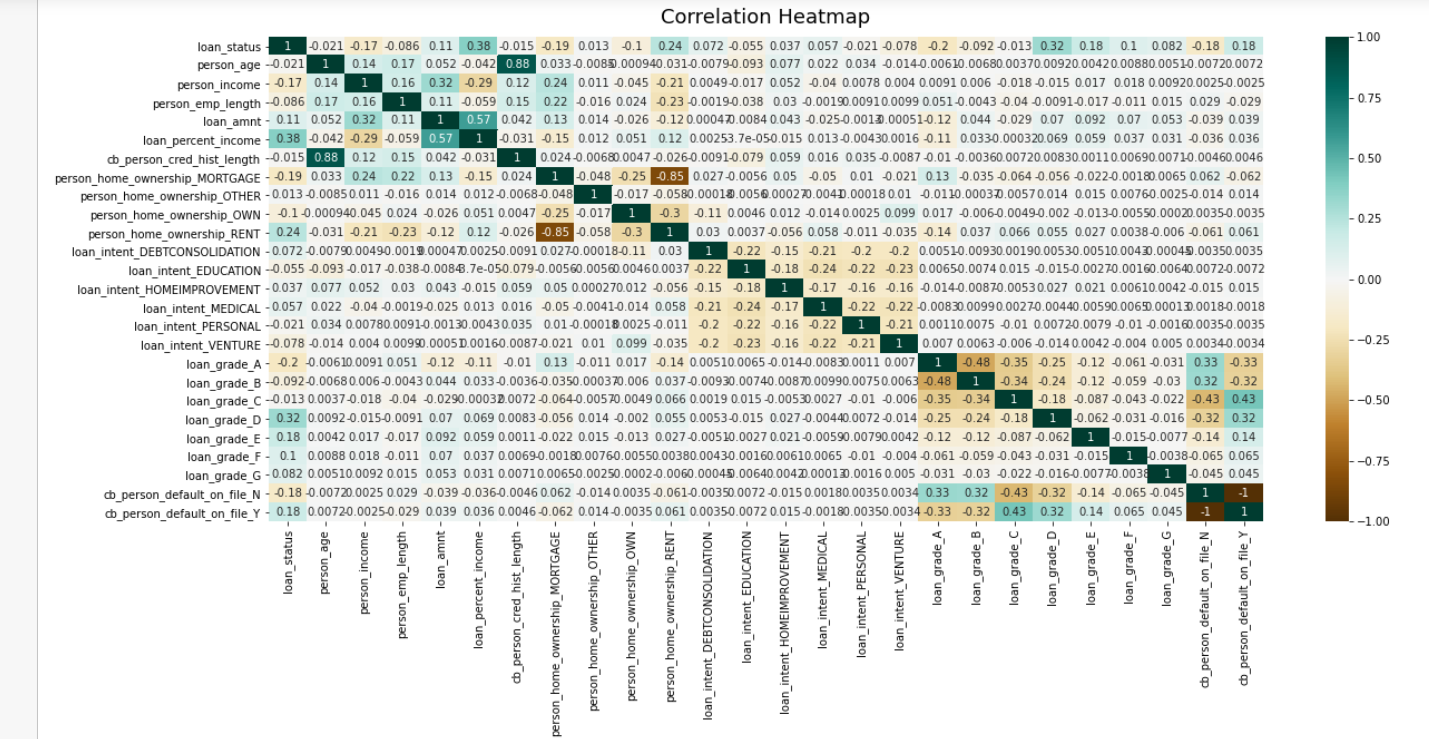




**We combine the numeric and one -hot encoded columns**



1. **Let’s the see ‘correlation between features”**



Multicollinearity :

**features strongly correlated to each other**

1. Person's age and db\_person\_cred\_hist\_length are strongly correlated. Correlation value is =0.88

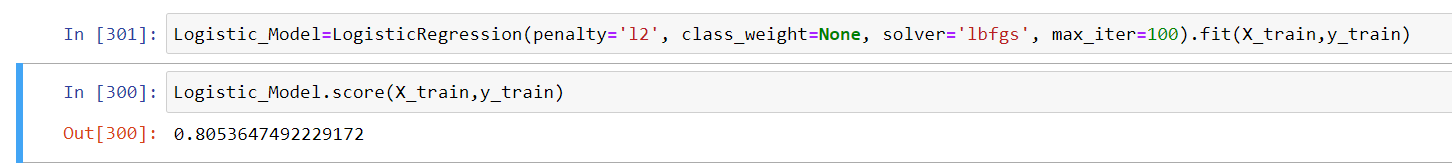
So we can drop either of it. Lets drop a column - db\_person\_cred\_hist\_length

1. similarly person\_home\_ownership\_MORTGAGE and person\_home\_ownership\_rent has strong -ve correlation between them

Let’s drop a feature - person\_home\_ownership\_MORTGAGE

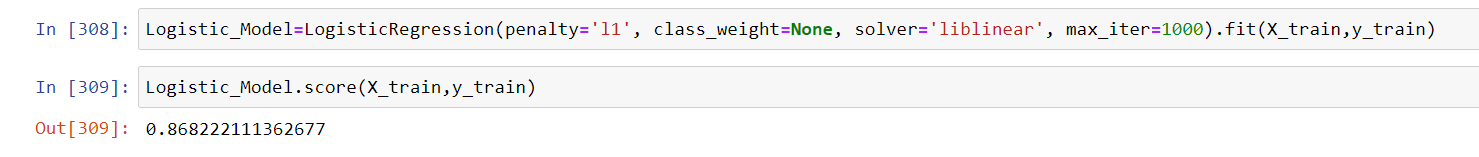
**Why - we won’t get unique solution if we keep highly correlated features in a model.**

**Let’s apply the model – let’s start with logistic regression**



**Accuracy on training data = 0.80**

**Let’s use L1 – which is Lasso regularization – which shrink the feature parameters to zero too if they are not important**

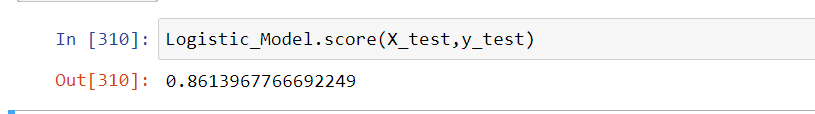


**Training score:- 0.87%**

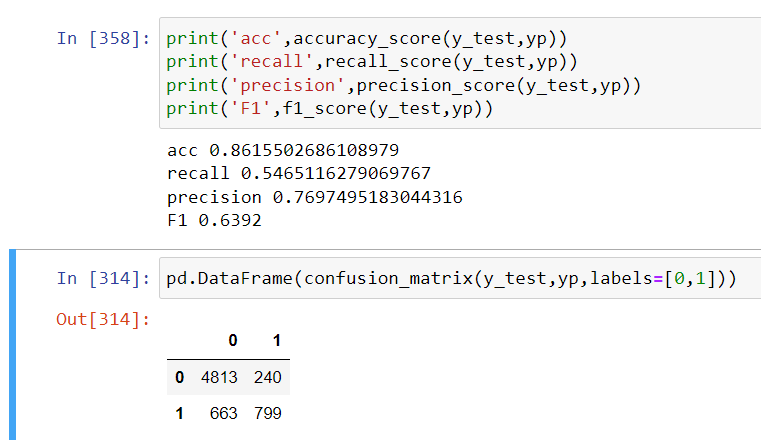
**I get better accuracy using “Lasso regularization” for Logistic regression model.**

**Validation :-**

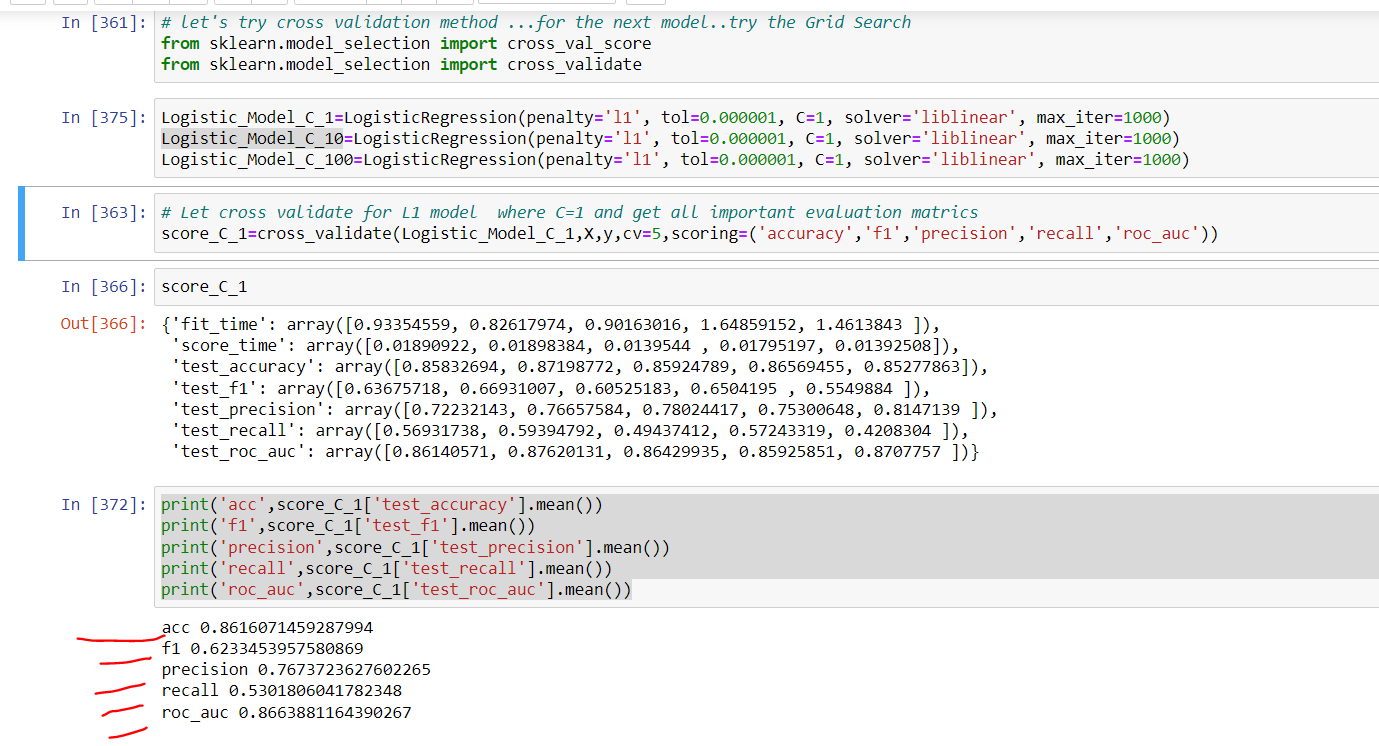
**Let’s test this on unseen data- which is test data set.**

 **Accuracy is 0.86 on a Test Data, which I think is very good.**

**However, other performance matrices is mix bag of performance result. Let’s apply other models and see what we see .**



**Let’s use Cross Validation to get an accuracy.**



**Let’s start second Model – KNN.**

**Here I checked the model by changing the hyperparameter values**

* No. of neighbours – I tried 5,10, 15, 30 – I do get better value for 10.
* Distance :- I also tested with hyperparameter for distance, - Manhattan distance, which is defined by p=1 and Euclidean distance which is defined p=2
* Accuracy is better for “number of neighbours =10 , P=1 ( which Manhattan distance)



**Let’s test the accuracy using Cross validation, also let’s consider the other performance metrics too.** 

As of now….between KNN and Logistic regresson – I see logistics has provided better performace matrics.

**Let’s start with Decision Tree now :- Let’s use the “Grid Search Method” to determine the best “model hyperparameters”**

Based on ‘grid search’ – the best hyperparameters are as follow

* Criteria – entropy
* Max depth =14, max\_features – sqrt, min\_sample\_splits=8



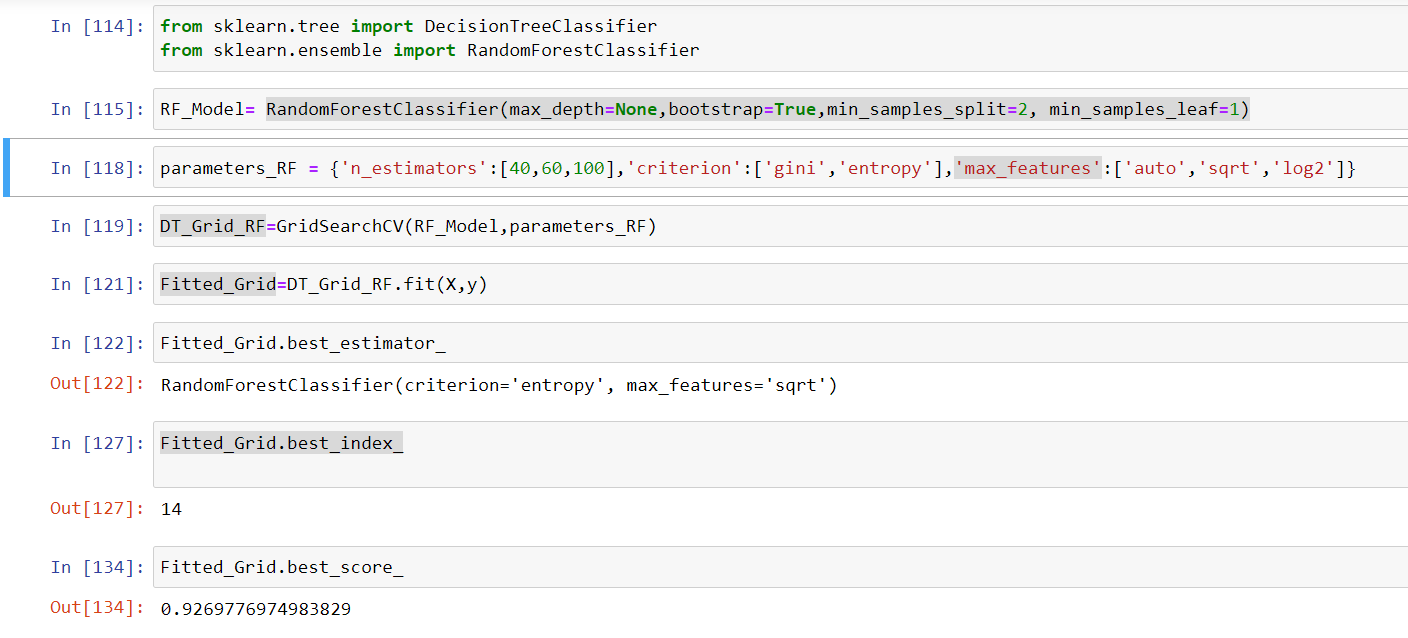
**Now, let’s calculate the other performance metrics for the best estimator**



**Let’s start with Random Forest Now – which is Bagging + Random selection of features.**

Since we are using Random foreset – which is mix of Bagging + decorrelation of features,

I think that we don’t need to have cross validation separately. By selecting hyperparameter, bootstrap=True for the random forest, we are using Bootstrap- which is Sampling with replacement.



**So as per Grid search the best hyper parameters for Random Forest are –**

**Criteria – Entropy**

**Max features = sq root.**

**I’ve kept the “Bootstrap = true”**

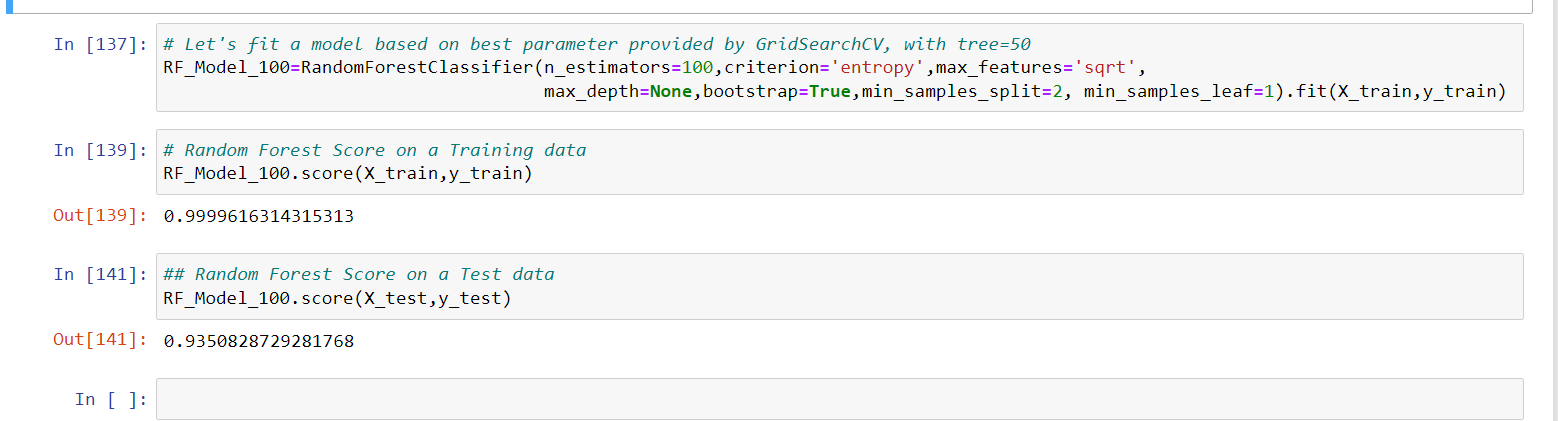
**Let’s create couple of Random Forest Models**

1. **Random forest with 50 trees**

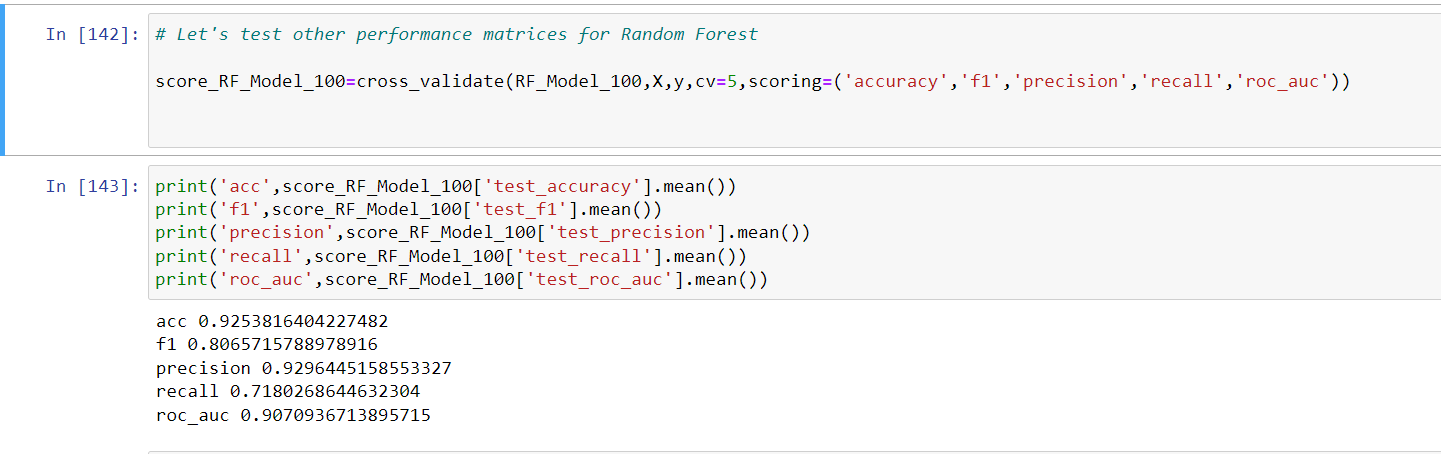
**Training accuracy =0.99, test accuracy = 0.93**



1. **Random Forest with 100 tress**

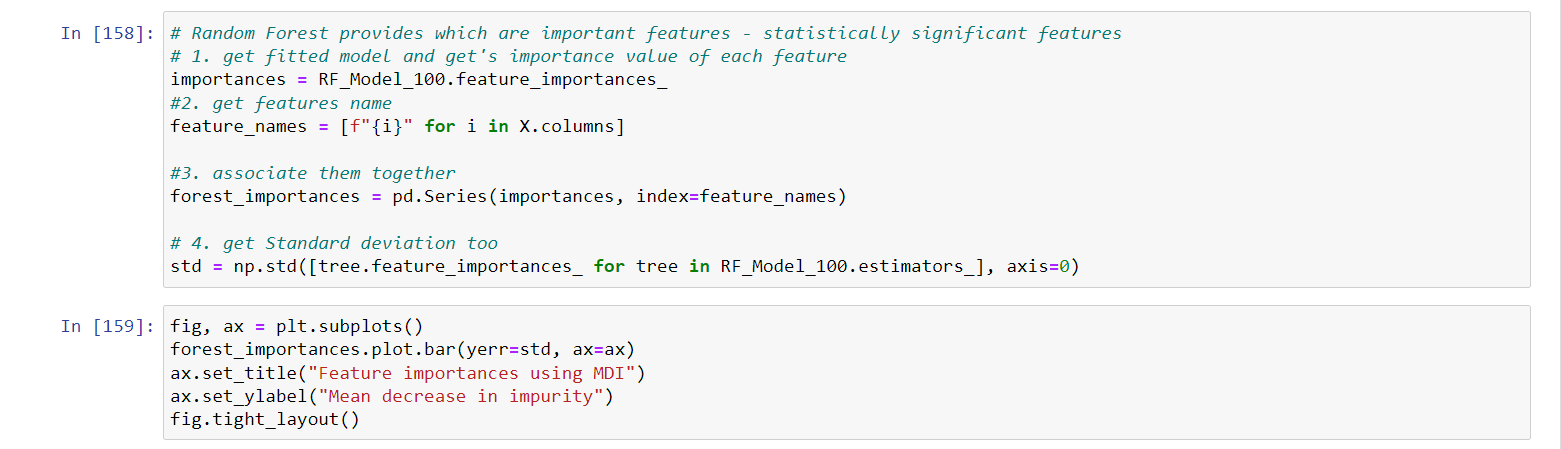


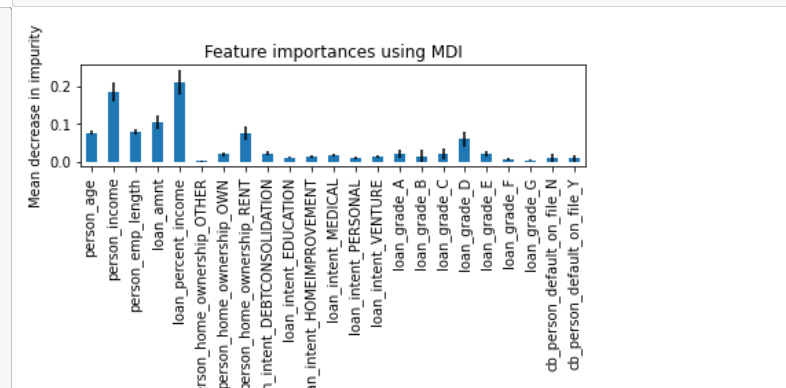
1. **Let’s test other performance matrices using “cross validating module”**



**We see significantly better performance values for Random forest compared to all of the previous models.**

1. **Let’s see which are features are important using Random Forest**





**So we can identify that person\_age, person income, person emp length, loan amt and loan percentage income are important features in determining if person will be default**

**Let’s start with Boosting**

* **Let’s start with Adaboost – where we use the same data with modification, as we assign weight to data based whether a record has been classified correctly or not**
* 

**Accuracy is better than decision free but little less than Random forest.**

**Let’ go with next model – Gradient Boost**



**Performance matrices for Gradient Boost are as good as Decision tree. Decision tree is slightly better though.**

**Summary**

* **Model tested for this classification problems are**

1. Logistic regression
2. KNN
3. Decision Tree
4. Random Forest
5. Adaboost
6. Gradient Descent Boosting

* **Methods/module used to get best parameters - GridSearchCV**
* **Method/module used to get all performance metrics together – Cross\_Validation**
* Random forest offers best performance metrics compared all other models
* Gradient boosting too has performance metrics as close as that of Random forest. There is not much difference in the performance metrices offered by “Random Forest” and “Gradient Boost”
* So based on overall performance metrics value ( Accuracy, F1, ROC\_AUC, Precision), we will use “Random Forest” to predict if we should provide loan to the prospective customer.

