2) Report Test Accuracy, Val Accuracy on each fold
Follow following format

FOLD	ACCURACY VAL	TEST
1	0.917391	0.917653
2	0.917519	0.91749
3	0.917442	0.917347
4	0.917646	0.918081
5	0.917621	0.917775
AVG	0.917524	0.917669

3) Report Visualization

First : A pdf report with following explanation.

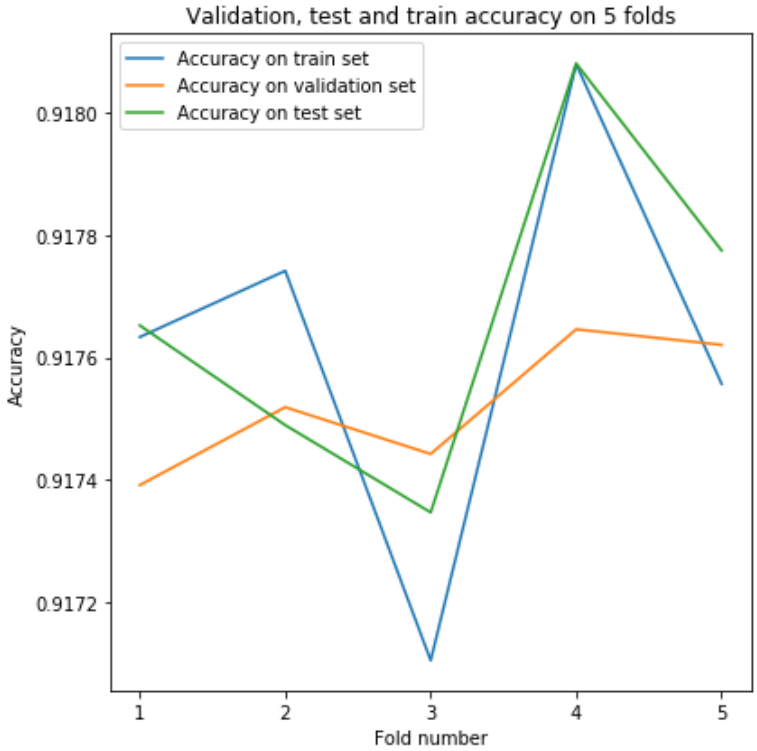
- What tools you used and why?
  To implement logistic regression was used next tools**:**

1. Batch gradient descent for searching best parameters by minimizing negative log-loss, because this is convex function.
2. Numpy library for computations, because it allow make quick and effective calculations (e.g. matrix, vectors multiplication).
3. Sklearn library for computing metrics of model's quality - confusion matrix, precision, recall, f1 scores, roc curves. All these metrics can show, is model better than random classifier, how can we trust predictions (for example, model can predict negative class very precisely, but be less accurate in prediction positive class (or vice versa). In this case total accuracy score will be pretty high, but f1 score - low.
4. Matplotlib and seaborn, because these libraries allow make very good plots.

- Metrics as explained in (2)

Fold 1
a) Val 0.917391
b) Test 0.917653
Fold 2
a) Val 0.917391
b) Test 0.91749
Fold 3
a) Val 0.917442
b) Test 0.917347
Fold 4
a) Val 0.917646
b) Test 0.918081
Fold 5
a) Val 0.917621
b) Test 0.917669
The average of 5 folds
a) Average of Val 0.917524
b) Average of Test 0.917669

- Visualization/Graph

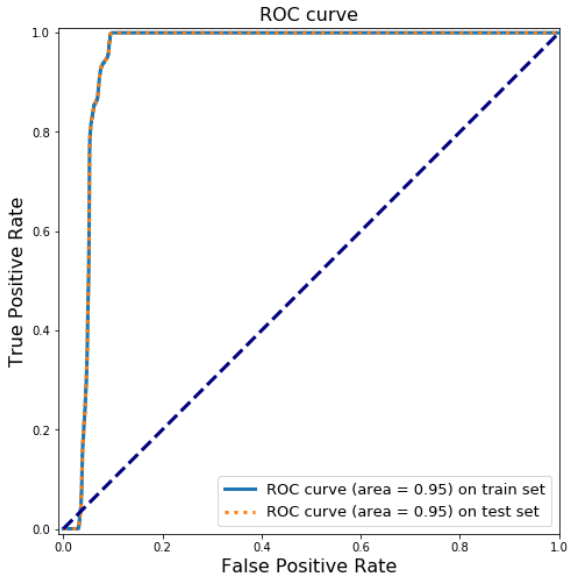


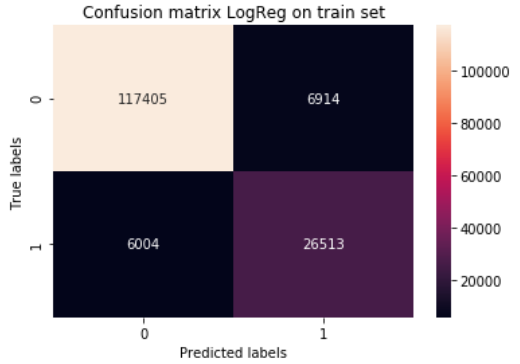
We can see, that logistic regression model can achieve accuracy of more than 91 percent. The accuracy of Val and Test of each fold is close. For example, we can see that in fold 1 the accuracy of Val 0.917391 and the accuracy of Test 0.917653 are close together.

- Conclude your experiment

To test model lets train it on full train set and again test on test set. It will give possibility to check if increasing of amount of data on training phase can improve model performance.

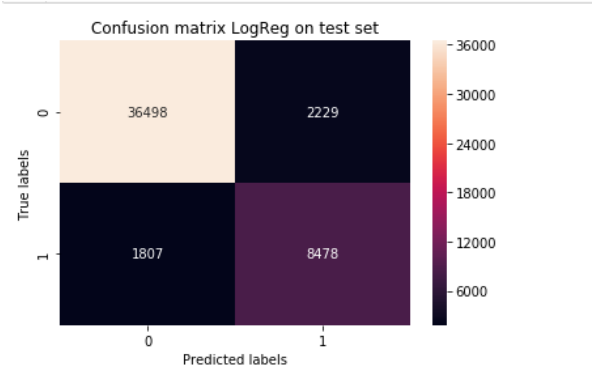
As a result accuracy on train set is 0.9176337065469663, test accuracy is 0.9176528197176 20. These both values are pretty high, but can be not enough informative. Let look on roc-au c curve, confusion matrixes of train and test sets and classification reports:





Classification report on train set

	precision	recall	f1-score	support
0.0	0.95	0.94	0.95	124319
1.0	0.79	0.82	0.80	32517
accuracy			0.92	156836
macro avg	0.87	0.88	0.88	156836
weighted avg	0.92	0.92	0.92	156836



Classification report on test set

	precision	recall	f1-score	support
0	0.95	0.94	0.95	38727
1	0.79	0.82	0.81	10285
accuracy			0.92	49012
macro avg	0.87	0.88	0.88	49012
weighted avg	0.92	0.92	0.92	49012

Model get high auc score, but if to see on classification report, precision, recall and f1-scores are lower than accuracy score. This fact gives possibility to say, that model with high insurance cannot to determine skin. So, if model classify instance as skin, only 79 percent chance that it is really skin.
For further work, I would add some more features, which maybe can better explain target value and help to make model more precise.

- Add a github repo as report

  https://github.com/weaf34/assignment2

Second : A github repo

https://github.com/weaf34/assignment2