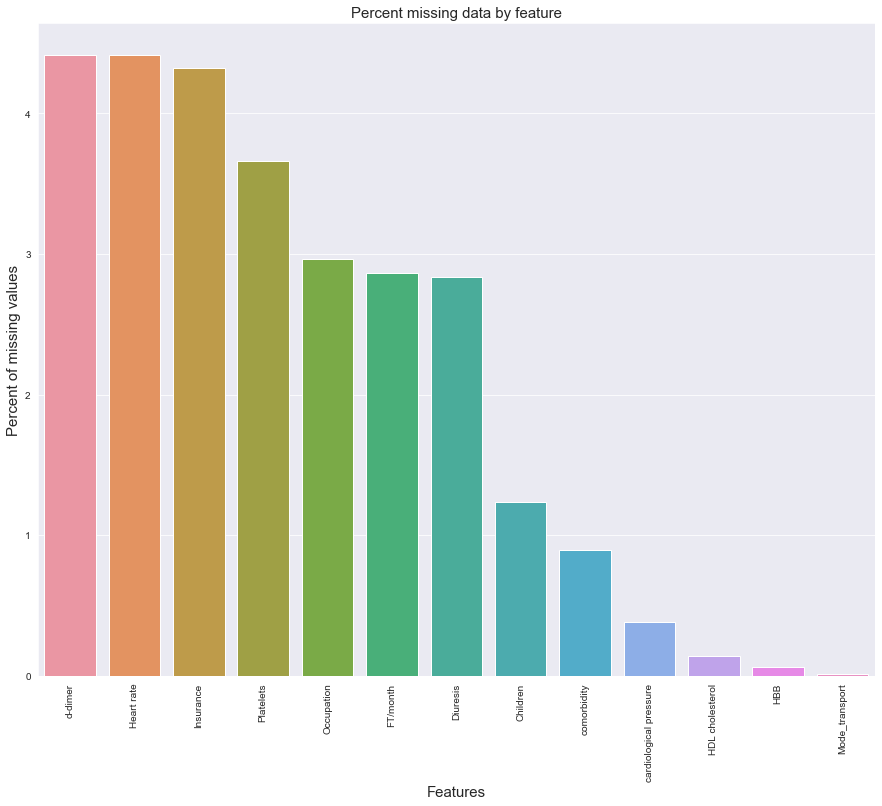
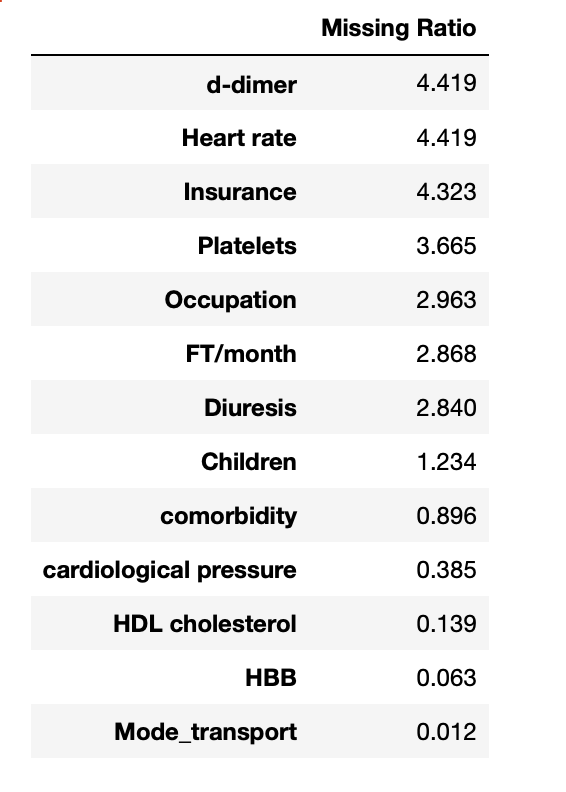
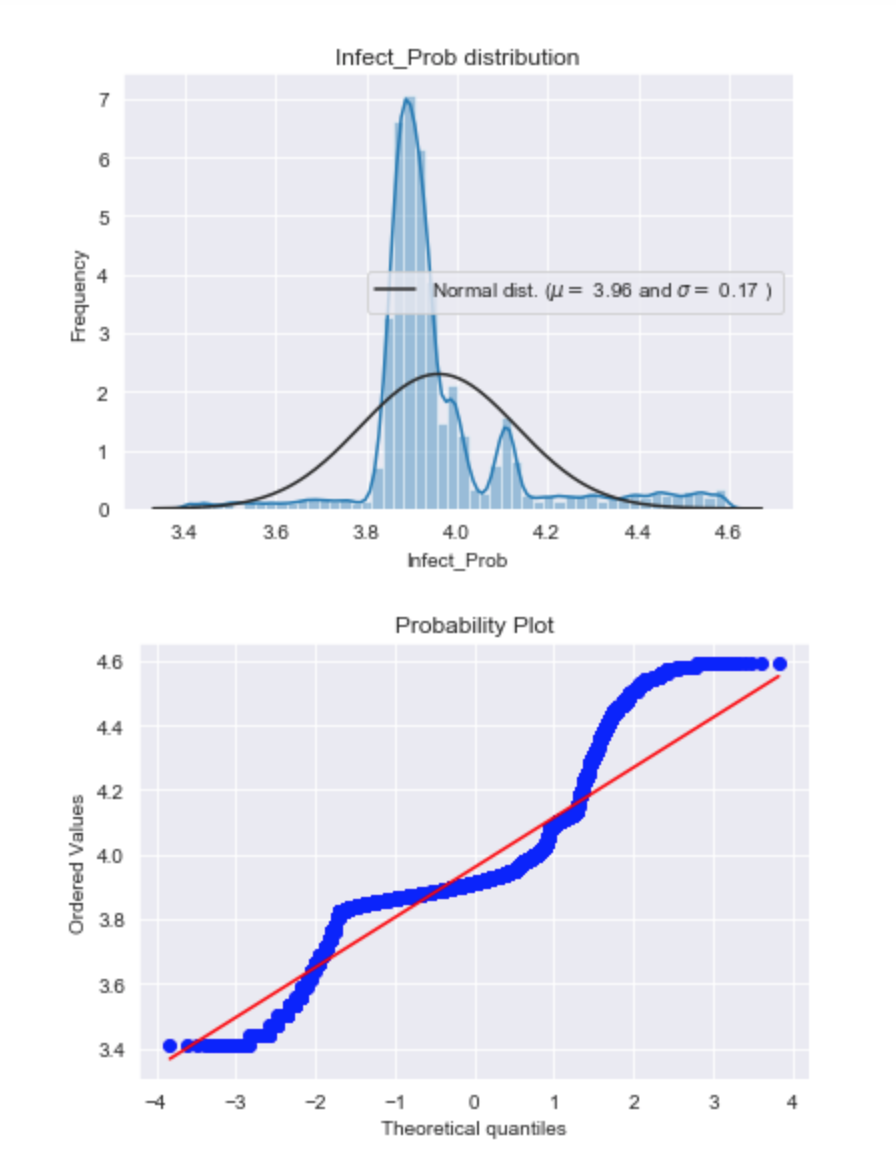
**Solution Sheet**

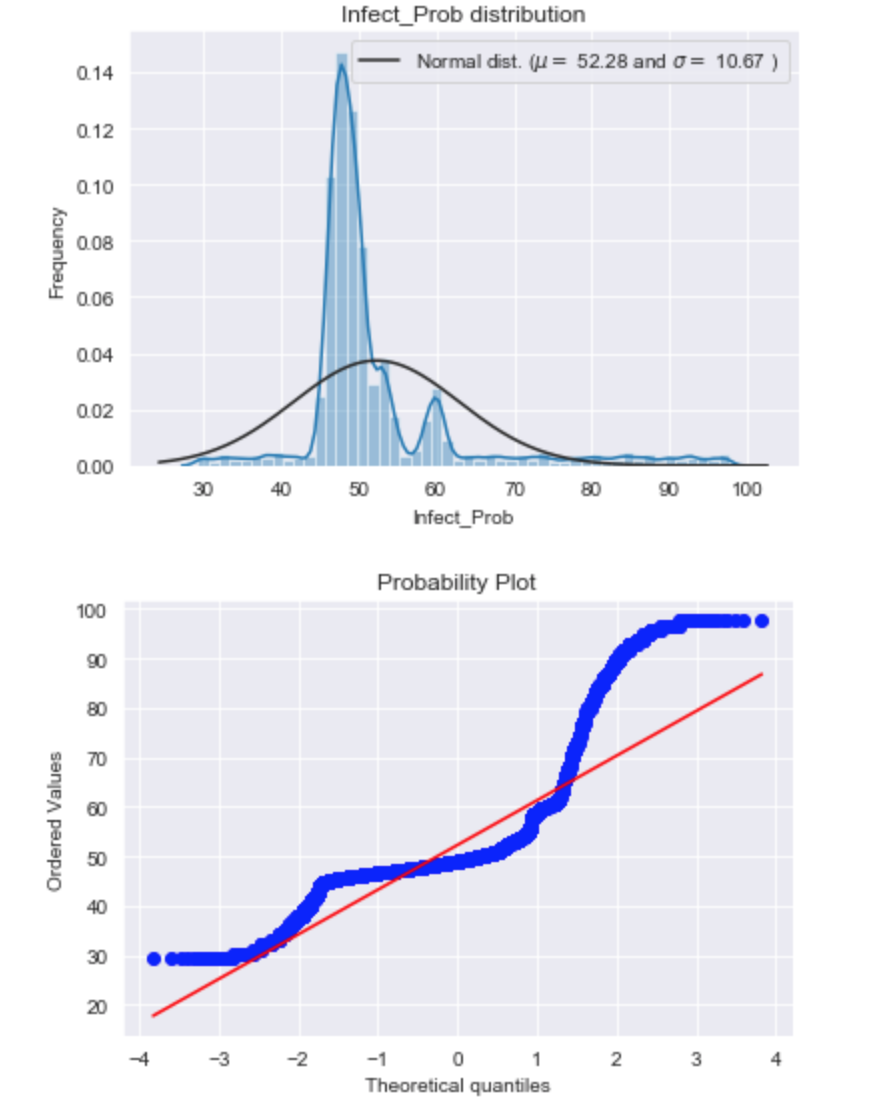
1. Which model have you used for probability prediction? Explain your model.

First features engeneering in done on the given data .**Imputing missing values** by proceeding sequentially through the data. .



The target variable is right skewed. As (linear) modelsworks well with normally distributed data , we need to transform this variable and make it more normally distributed

Log transformation is done on the target variable. use the numpy fuction log1p which applies log(1+x) to all elements of the column

still the skew is not correct but its better

Then we choose many base models (mostly sklearn based models + sklearn API of DMLC's XGBoost and Microsoft's LightGBM), cross-validate them on the data before stacking/ensembling them. The key here is to make the (linear) models robust to outliers. This improved the result both on LB and cross-validation.

BASE MODELS

* **LASSO Regression** :

This model may be very sensitive to outliers. So we need to made it more robust on them. For that we use the sklearn's **Robustscaler()** method on pipeline

* **Elastic Net Regression** :

again made robust to outliers

* **Gradient Boosting Regression** :

With **huber** loss that makes it robust to outliers

**XGBoost** :

**LightGBM** :

BASE MODEL SCORES

Lasso score: 0.1513 (0.0068)

ElasticNet score: 0.1513 (0.0068)

Kernel Ridge score: 0.5737 (0.0195)

Gradient Boosting score: 0.1557 (0.0046)

Xgboost score: 0.1551 (0.0036)

LGBM score: 0.1477 (0.0054)

Less simple Stacking : Adding a Meta-model[¶](https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard/notebook#Less-simple-Stacking-:-Adding-a-Meta-model)

In this approach, we add a meta-model on averaged base models and use the out-of-folds predictions of these base models to train our meta-model.

The procedure, for the training part, may be described as follows:

1. Split the total training set into two disjoint sets (here **train** and .**holdout** )
2. Train several base models on the first part (**train**)
3. Test these base models on the second part (**holdout**)
4. Use the predictions from 3) (called out-of-folds predictions) as the inputs, and the correct responses (target variable) as the outputs to train a higher level learner called **meta-model**.

The first three steps are done iteratively . If we take for example a 5-fold stacking , we first split the training data into 5 folds. Then we will do 5 iterations. In each iteration, we train every base model on 4 folds and predict on the remaining fold (holdout fold).

So, we will be sure, after 5 iterations , that the entire data is used to get out-of-folds predictions that we will then use as new feature to train our meta-model in the step 4.

For the prediction part , We average the predictions of all base models on the test data and used them as **meta-features** on which, the final prediction is done with the meta-model.

To make the two approaches comparable (by using the same number of models) , we just average **Enet KRR and Gboost**, then we add **lasso as meta-model**.