Triaging Software Bugs

Raphael Matile Yves Steiner

Big Data and Business Analytics

December 15, 2016

Introduction

Steps considered in the process of creating the prediction model

- System setup and technologies
- Preparing the data
- Considered prediction variables
- Results of regressions
- Conclusion

System setup/technologies

Used technologies to create this model:

- Python3.5
- Numpy
- SQLite3
- scikit-learn (linear models)
- Keras and Theano (neural net)

Dataset

Mozilla and Eclipse Defect Tracking Dataset (MSR 2013)[1]

Contains information about:

- Current status
- Assigned Priority
- ...

Includes incremental modifications of each bug

[1] https://github.com/ansymo/msr2013-bug_dataset

Preparing the data

The data was derived from the two CSV files required some reformatting in order to import them into the database. The `CC` and the `short-desc` file.

Preparing the data - CC-file

The CC file had multiple items separated by a comma for the `what` column, this resulted in a file reading error since it created more than the expected four columns. We used the following code to format the CSV file:

```
for row in reader:
    length_row = len(row)
    str = row[1]
    if length_row > 4:
        for x in range(2, 2 + length_row - 4):
            str = str + ',' + row[x].strip()
        row[1] = str
        row[2] = row[2 + length_row - 4]
        row[3] = row[3 + length_row - 4]
        for x in range(0,length_row-4):
            del row[-1]
```

Preparing the data - Remaining Files

To import the CSV data into our database we used a regular expression, which captured the groups: id, what, timestamp, who

```
([0-9]+),((?:.|(?:\n\t)+)*),([0-9]+),([0-9]+)
```

Preparing the data - Training, Validation, Test Set

In order to create valid models, we set up the database with the given data and split it up into train (50%), validation (25%) and test (25%) sets.

The validation set is currently only used for training the neural network

Considered prediction variables[2]

1. Success Rate of a bug assignee
2. Success Rate of a bug reporter
3. Success Rate of a bug report for every reporter-assignee pair
4. Success Rate of a bug in terms of how many times it got reassigned
5. Success Rate for number of reassignments of a bug
6. The duration in seconds of how long a bug was opened
7. Success Rate of the component to which the bug was assigned
8. Success Rate of a bug considering the reporter and all names on the CC
9. Success Rate of the software version to which the bug was assigned

[2] These prediction variable are all chosen from the list of the use case.

Considered prediction variables - Success

In order to determine if a bug was successful or failed, we defined the following combinations of `current_status` and `current_resolution` for a report as a success:

current status	**current resolution**
RESOLVED	WORKSFORME
RESOLVED	FIXED
VERIFIED	FIXED
CLOSED	WORKSFORME
CLOSED	FIXED

Considered prediction variables - Fail

The following combinations of `current_status` and `current_resolution` are considered as a fail:

current status	**current resolution**
RESOLVED	WONTFIX
RESOLVED	INVALID
CLOSED	WONTFIX
CLOSED	INVALID
VERIFIED	WONTFIX
VERIFIED	INVALID

Results of regressions

After calculating the prediction variables we've run different prediction models in order to determine their performance:

Model	Accuracy	f1-Score
Classic linear regression	0.751108192415	0.836567485985
Lasso linear regression	0.731012969956	0.826300835418
Logistic regression	0.752290264324	0.836625086625
Bayesian ridge regression	0.751108192415	0.836574533224
Decision tree	0.907273025776	0.92358480355
Neural Netowrk	0.652963388606	0.790051846408
Optimized Neural network[3]	0.81037282411	0.81037282411

[3] Uses the first 120000 entries as training, the last 30000 as test set

ROC Curves

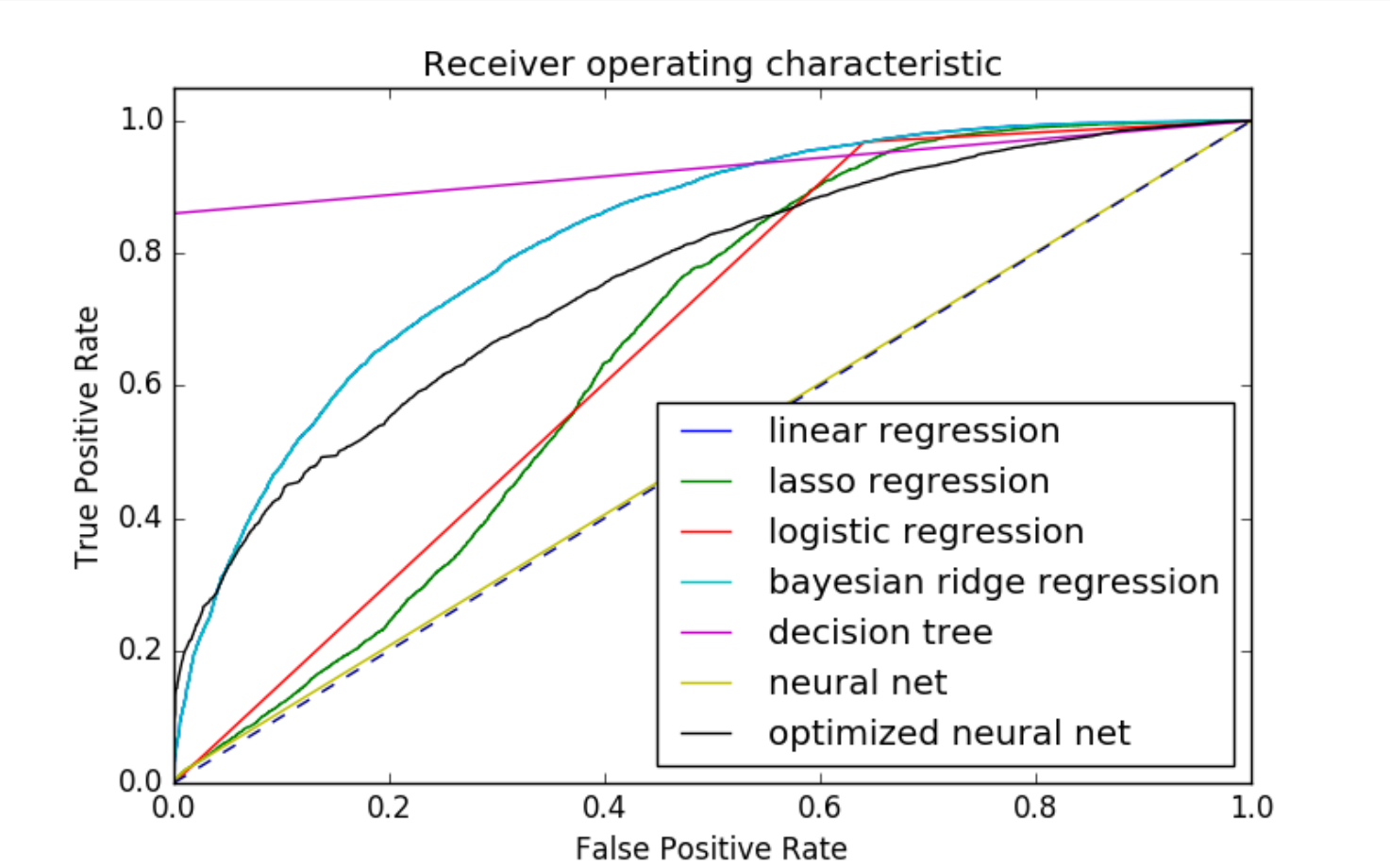

Figure: ROC Curves for different validated models

Conclusion

Surprisingly, the decision tree achieved the best score over all models. Looking at its structure[4], its dimensions can indicate a certain grade of overfitting to the data. However, cross-validation showed, that different folds result in completely different accuracies.

Also, the optimized neural network, which uses the first 120000 entries of the total set, indicates that there is a structure given in the dataset, resulting in quite different results, when shuffled.

[4] Included in the submitted code repository