BUGZY

Predict if a git commit is bug fix

Vivek - vvivek@ncsu.edu

Motivation and Background

- Identifying a bug fix through commit message
- Help in understanding features of defective code
  - Aid default predictors
- Automated prediction of bug fixes
  - Limited inter-project variation
- Natural Language Processing is FUN ! (and complex !)
- Aim:
  - Map existing domain knowledge with unsupervised learning
  - **Comprehensible**
  - **Stable**

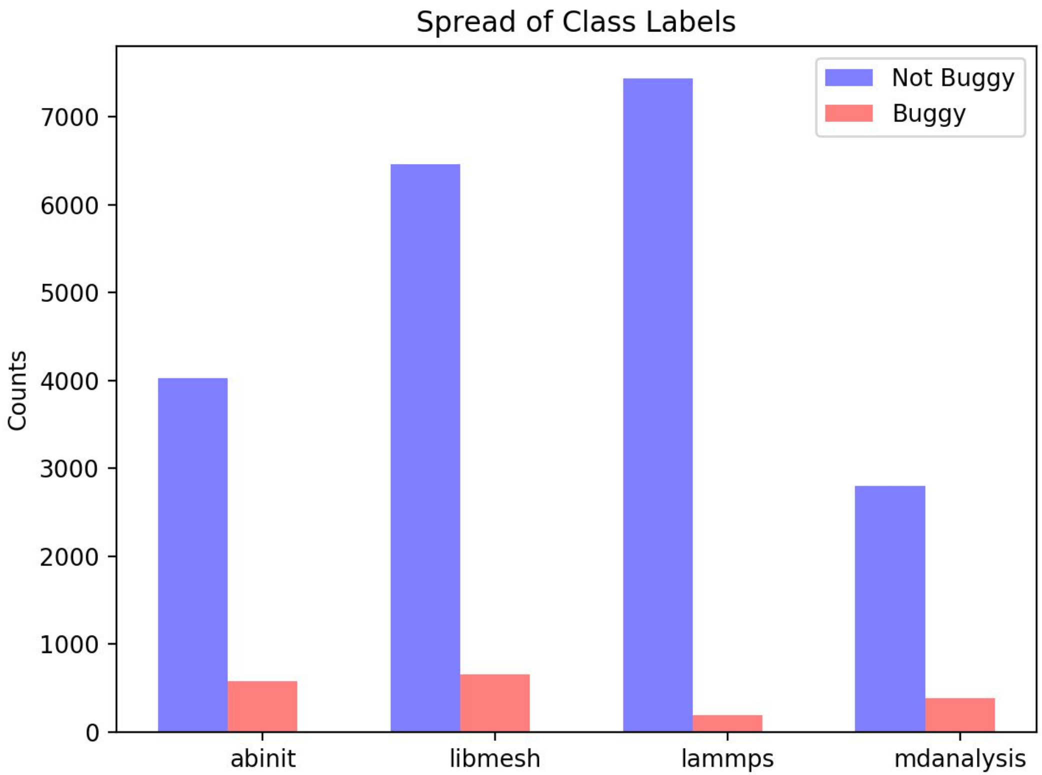
Dataset

- Open source software projects - “abinit”, “libmesh”, “lammps”, “mdanalysis”
- Labeled by mechanical turks in Hackathon
- Columns - Hash, Timestamp, Message, Buggy

	abinit	libmesh	lammps	mdanalysis
0 (Not Buggy)	4024	6462	7437	2802
1 (Buggy)	572	651	186	379

https://github.com/se4sci/data-collection/tree/master/labeled_commits/

Are there enough positive labels?

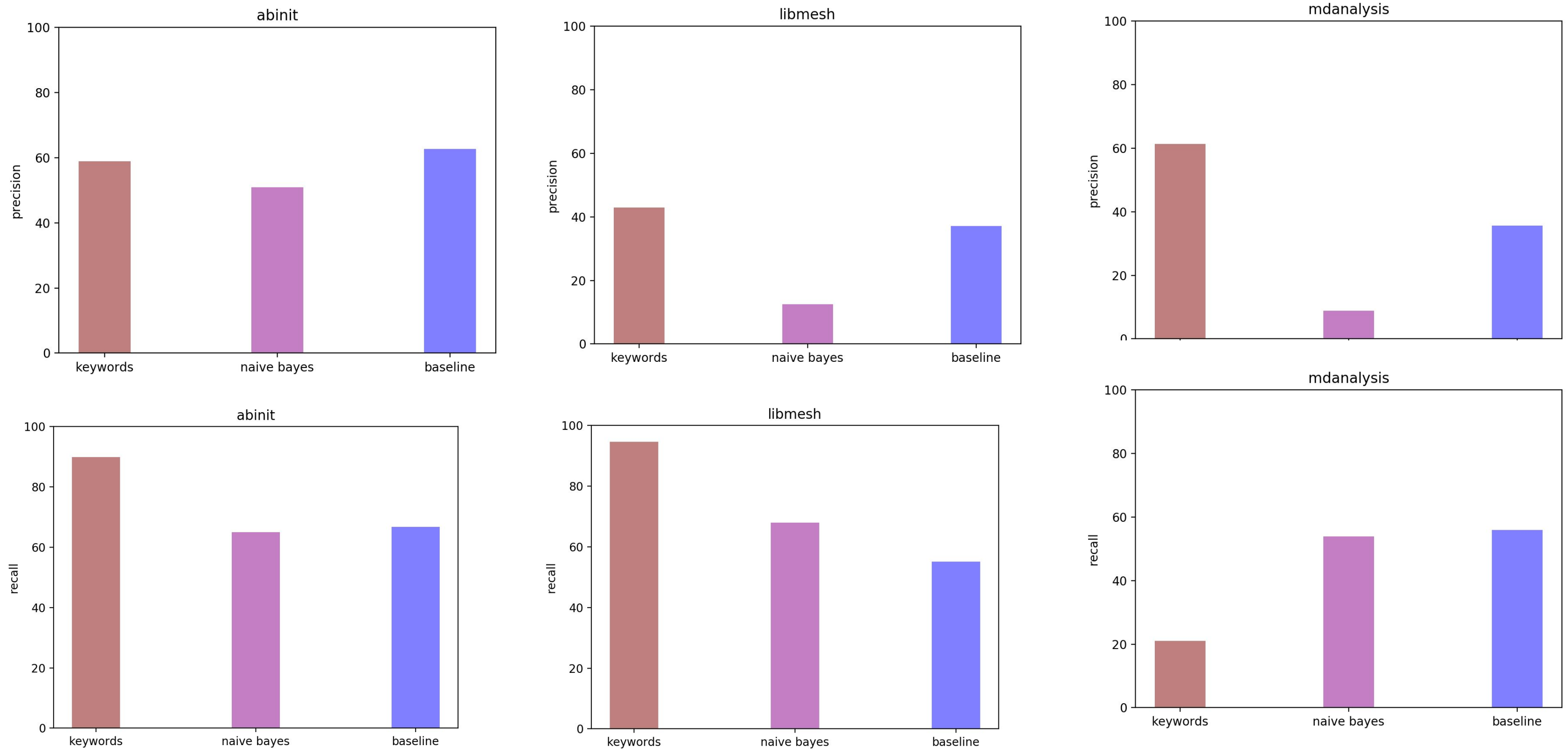


Baseline

- With Domain Knowledge as features
  - ["bug", "fix", "wrong", "error", "fail", "problem", "patch"]
- Multinomial Naive Bayes
  - Term frequency
- Support Vector Machine
  - TF-IDF weighted vectors
- Baseline is **TF-IDF+SVM**
  - More general, low variations among projects

Note: Ignoring LAMMPS dataset from SVM as FN and TP are zero

Baseline Metrics



Research Questions

- Presence of a word vs count of buggy words?
  - Should we use TF-IDF?
  - Should we use TF?
- Why only these words as buggy words?
  - Can these words change over time?
  - Over projects?
  - Can we explain why a word is buggy word?
- Can we create an automated machine learning model ?
  - Performs equivalently or better
  - Generalizes according to project

Topic Modeling

- Latent Dirichlet Allocation
  – Unsupervised clustering
- Create 2 topics
  – 2 class labels
  – Buggy and Not Buggy
  – Create from top 100 frequent words in vocabulary
- Extract top 10 words as features of a topic

Magic params
- max_features = 100
- Num_top_words = 10

Top 10 Words per Topic

BUGGY ?

	Topic 1	Topic 2
abinit	merge, branch, develop, abinit, trunk, remote, gitlab, tracking, release, org	**fix**, test, file, **update**, **add**, new, ref, **change**, typo, error
libmesh	merge, request, pull, libmesh, mesh, **update**, test, make, new, example	**fix**, **added**, **add**, use, element, **change**, file, function, **fixed**, class
lammps	svn, lammps, git, icms, trunk, temple, edu, sync, library, **added**	**fix**, merge, request, pull, kokkos, user, pair, **update**, dpd, **add**
mdanalysis	merge, doc, mdanalysis, develop, pull, request, branch, code, atomgroup, issue	test, **added**, **fixed**, **fix**, issue, **updated**, file, analysis, **changelog**, new

["bug", "fix", "wrong", "error", "fail", "problem", "patch"]

Agreement of LDA with ground truth?

- Recall
  - Of all the True labels, how many do I recall?
  - True Positive / (True Positive + False Negative)

	Recall (%)
abinit	96.85
libmesh	79.72
lammps	100.0
mdanalysis	88.39

- LDA can recall most of the labels if we consider Topic 2 as buggy topic based on domain knowledge

What about Precision? A classifier?

Precision:

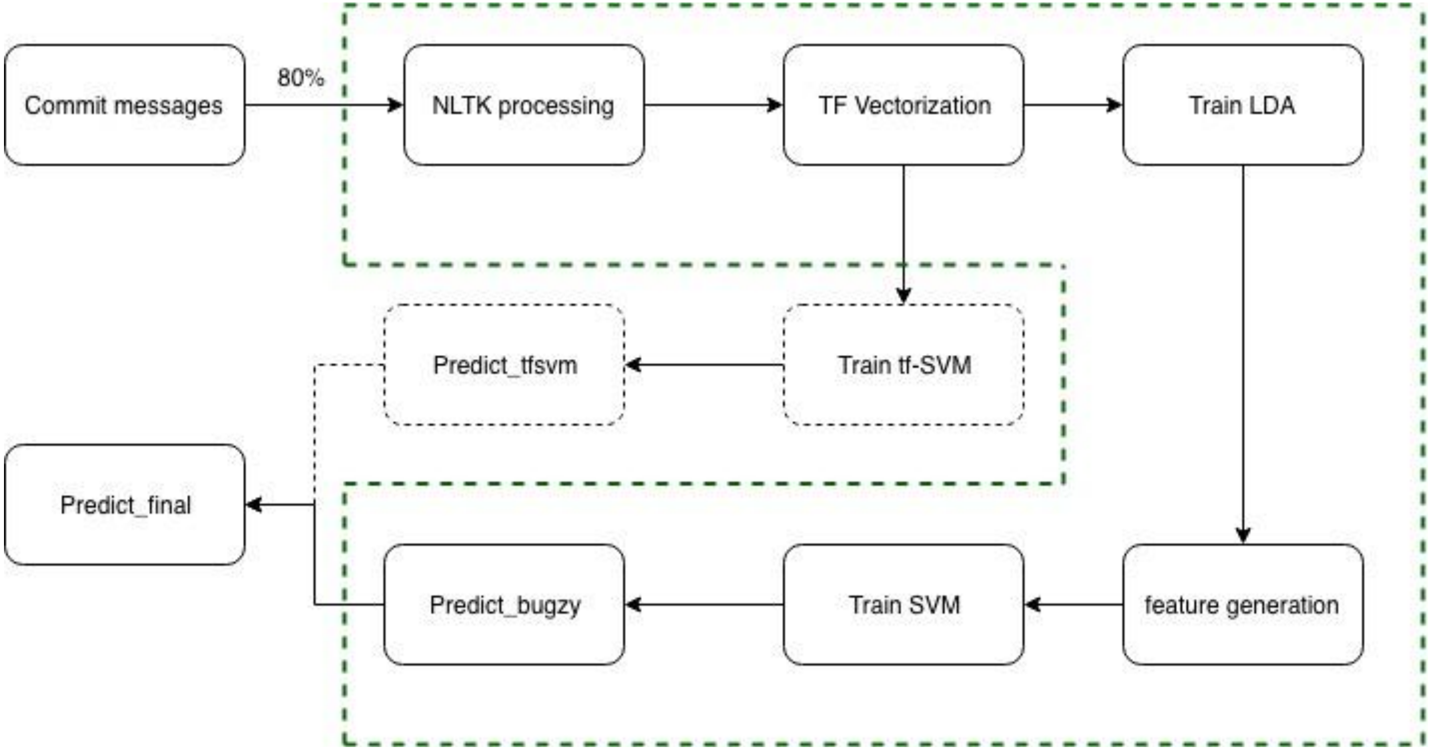
- Everything that I say is buggy, how many actually are?
  - True Positive / (True Positive + False Positive)
- Ideally,
  - less false positives (reduce false alarms)
  - false negatives (predict a bug-fix with high confidence)
- Train a classifier that uses LDA topic probabilities
- Support Vector Machine + LDA i.e. **BUGZY**
- Create features based on words characterizing a topic
- So, SVM + (LDA transformed features)

BUGZY

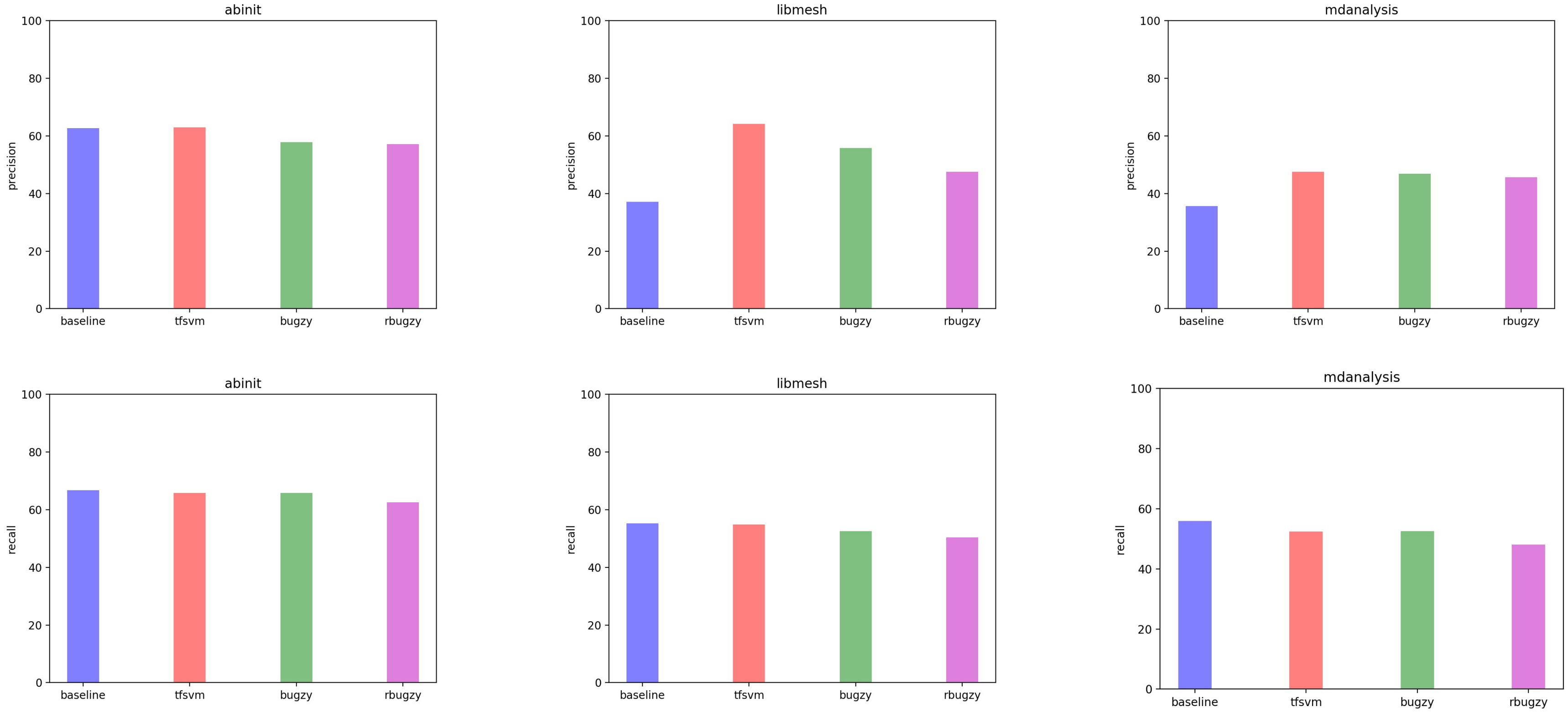
- Use 80 percent of raw data for training and validation
- LDA - train on tf-vectorized words
- Extract top 10 word W = (W_Topic_1 union W_Topic_2)
- For each commit, for each word in W
  - Compute count * lda_probability
  - Append LDA topic probability
- Shape is num_rows* 22
- Train SVM and Validate
  - R-BUGZY - train Random Forest instead SVM

BUGZY helps developers sleep, as HUGSY helped Joey.

BUGZY - workflow



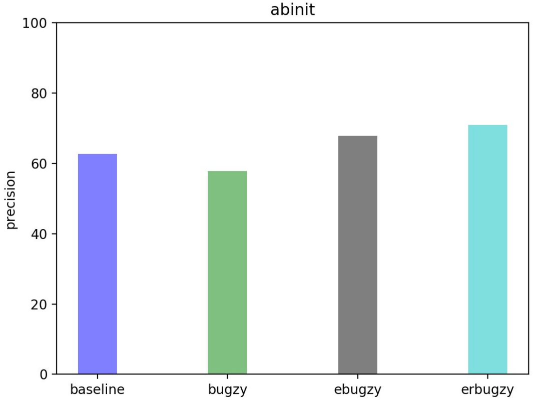
BUGZY Metrics

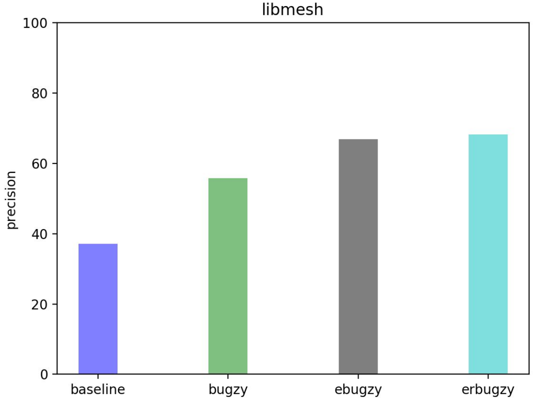


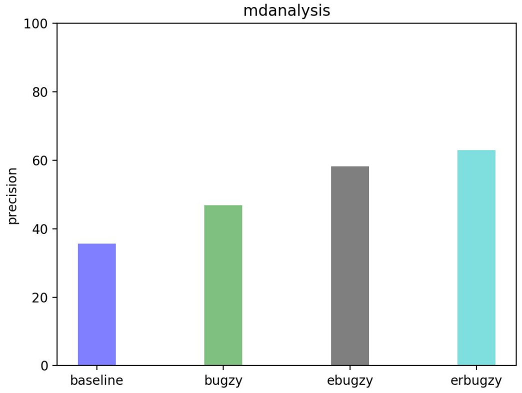
Ensemble BUGZY

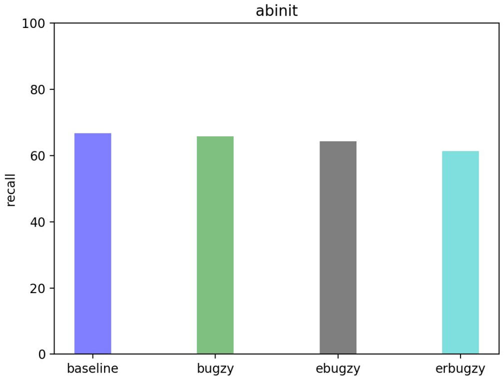
- Baseline SVM performance almost equivalent
- But LDA recall was high
- (BUGZY) or (TF+SVM)
  - If any of them say yes buggy, classify as buggy
  - Else, not buggy
- Does LDA capture more features?
  - Combine R-BUGZY and TF+SVM

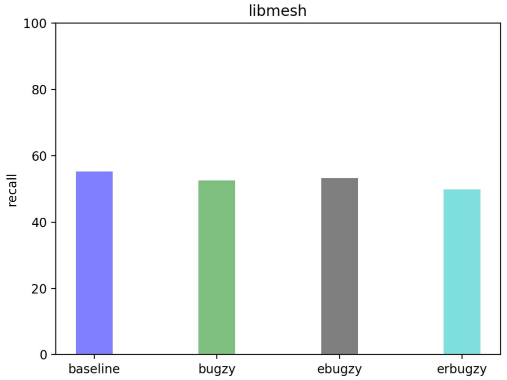
E-BUGZY Metrics

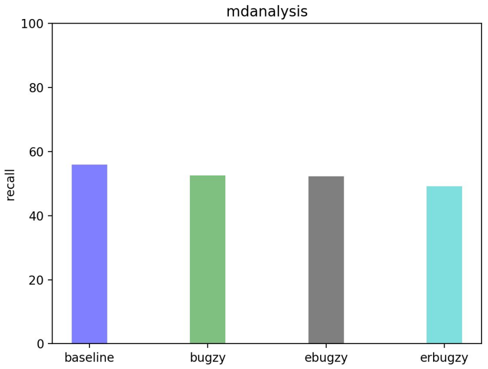












Conclusions

- TF makes more sense
- Using LDA makes SVM more **comprehensible**
  - Without much loss of precision or recall
  - Was the model reasonable?
  - Can I now explain why a buggy word is called buggy?
  - Can I now find more buggy words with confidence?
- E-BUGZY, ER-BUGZY
  - No wild variations over projects or random data
  - Potential to generalize over projects
  - **Stable**

Limitations

- Dataset
  - More projects, more well maintained projects
- Statistical tests for significance and effect size
- Why ‘lammps’ metrics are bad with SVM?
  - Works with probabilistic model such as NB
- Group similar words with software engineering context
  - added, add, adds, enhance, incremental

Future Work

- LDA has very good recall, how can that be used to train SVM better?
- Feature encoding
  - Word2Vec?
  - GloVe?
- Train on well maintained software projects
- Can this model be generalized?
  - Train on few projects and test on others
  - Temporal?

Thank you!

?