*

General Theories of Software Defect Prediction: A Preliminary Report

William Mensah
WVU, Morgantown, WV
wmensah@csee.wvu.edu

Adam Nelson
WVU, Morgantown, WV
anelson8@csee.wvu.edu

Tomi Prifti
WVU, Morgantown, WV
tprifi@csee.wvu.edu

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings. It is an *alternate* style which produces a *tighter-looking* paper and was designed in response to concerns expressed, by authors, over page-budgets. It complements the document *Author's (Alternate) Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file. A compiled PDF version is available on the web page to help you with the 'look and feel'.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Delphi theory

Keywords

ACM proceedings, LaTeX, text tagging

[*]A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX* at `www.acm.org/eaddress.htm`

1. INTRODUCTION

By predicting defects in software systems *before* the deployment of that software, it is possible to gauge not only the probable quality upon delivery, but also the maintenance effort. Software defect prediction builds models using available company data that can then be applied in order to predict these software faults. But in order to employ these models, a company must have a data repository where information regarding defects from past projects are stored. However, according to Turhan, et. al. [7], few companies are applying this practice. Turhan, et. al. claims (and we agree), that this is most likely due to a lack of local data repositories. When this is the case, companies must use non-local data in order to build defect predictors. Thus, it is not only important to determine how well cross-company data can be used when local data is unavailable, but also to find the presence or absence of a general theory of software defect prediction. In other words, in determining the existence of an empirically-derived correlation between varying defect data sets, a statement can be made in regards to not only the stability of current defect prediction, but also the underlying similarities of cross-company software projects. On the other hand, if no correlation is found to exist, instability of those current predictors may suggest that further research should be conducted in order to provide incite into the variance between projects.

2. BACKGROUND

The ability of an organization being able to use cross-company (CC) data when within-company (WC) data is not available in order to build defect predictors would be advantageous. However, it remains unclear if this practice can yield beneficial results.

Turhan et al. conducted three experiments to rule in favor of CC data obtained from other sites, or WC data gathered locally. The conclusions of those experiments show that CC data, when applied using *relevancy filtering* via a k-nearest neighbor scheme. The idea behind the k-NN filter is simple; by building a training set that is homogeneous with the testing set, it is assumed that a bias in the model will be introduced. The filter works as follows: for each instance in the test set, the k nearest neighbors in the training set are chosen. Then, duplicates are removed and the remaining instances are used as the new training set. This relevancy filtering can lead to defect predictors almost as effective as WC data. Thus, as stated by Gay et. al. [5], "...while local data is the preferred option, it is feasible to use imported data provided that it is selected by a relevancy filter."

Gat et. al. confirmed Turhan et. al.'s results, but instead of implementing a nearest neighbor filter, a locally weighted scheme was used to filter the data via [4]. This experiment was of significance due to the fact that Gay et. al.'s results showed not only that CC data can be used when local data is not available, but also that publicly available data such as the PROMISE [1] data.

On the other hand, [1] shows that CC data cannot be used to build accurate defect predictors. For example, in one experiment conducted by Zimmerman et. al., Firefox and Internet Explorer were used due to their domain relationship (browsers) in order to determine how well one could predict for the other. It was found that while Firefox could predict for Internet Explorer at a precision of 76.47%, Firefox could *not* predict for Internet Explorer approaching the same precision (4.12%). However, this experiment did not utilize any form of relevancy filtering, so it is unknown how the two data sets would react to predicting for one another under these circumstances.

If CC data, when filtered, can be used in order to predict defects, we are left to assume that this is due to the fact that the data sets share some innate similarities of their metrics. But if it is found conclusively that CC data cannot most generally build good defect predictors, we are to conclude that other measures must be taken in either the collection of the WC data, or for more research in the further filtering of CC data.

2.1 Defect Prediction Stability

3. THE EXPERIMENT

3.1 Preparing the Data

We employed a number of preprocessing techniques to prepare the original data before applying our learner to it. These include:

- Logging
  Replacing all number values N with log(N) when N > 0.0001 or with 0.0001 if otherwise.
- Binning
  Splitting each data set into *10* bins (by default) and then building train data from 90% of the data and 10% from what is left.
- Clustering
  Partitioning *n* observations from each data set into k clusters using *k-means* algorithm whereby each observation belongs to the cluster with the nearest mean.
- Feature Subset Selection
  Sample bias via *InfoGain* whereby each column in the data set is ranked by some criteria and the top n columns are maintained and used for the experiment because they are more likely to have a stronger influence on the results.

Why are all these preprocessing techniques necessary and why is time wasted them before the learner is applied? Why not use the data in its raw form? When it comes to data mining, real world data is considered by most software engineers as *dirty*. This is because the data could be incomplete, that is, be missing some attributes or attribute values or it could simply consist of only aggregated values. In addition to that, the data could be inconsistent, and could also contain errors and outliers.

3.2 Discretization techniques

Two discretization method were used for performing the experiments: Equal Interval Width (10 bins) and K-Means. Both methods fall under the unsupervized discretization methods class. Equal bin length is a global techinque which replaces each value by the identifier of the bin. The range of values is divided into k equally sized bins where k is a parameter supplied by the user. It makes no use of the instance class and is thus an unsupervized discretization method. Equal Bin Length is a global discretization method since it produces a mesh over the entire n-dimensional instance space [2]. The k-means on the other hand is a local discretization tool which groups the data into a number of clusters based on some similarity measure. Local discretization methods are applied to local regions of the data.K-Means is grouping the rows in k different clusters using the Eucledian distance as a measure of similarity between the instances. At the beginning the centroids (means) for each cluster are picked at random and then the instances are assigned to the group with the closest centroids to that particular instance. K-Means will stop iteration through the instance set when the centroids for all the clusters do not change anymore or when a stopping criteria is reached. In both cases of discretization the data is preprocessed by taking the log of each numeric value. Since these two method do not take into consideration class lables there might be some loss of classification information as a result of grouping instances that are stronly related but belong to different classes. Since our main learning algorithm is Naive-Bayes, both discretization method greatly improve the performance of the learner. Naive-Bayes is used with both k-means clustering and equal interval width bins on each of the datasets.

4. RESULTS

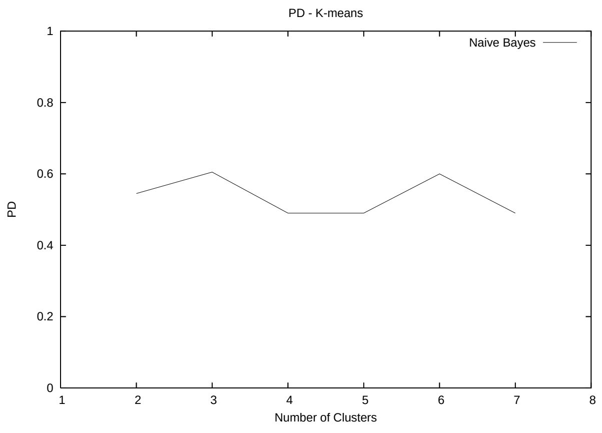
Naive-Bayes learner was applied on all data sets. Three learners were tested

- Naive-Bayes without discretizing. K-means applied on the data
  In this experiment the data was supplied to the learner without performing any discretization. After splitting the data the train set was clustered using k-means. For each instance on the test data set the closest cluster was determined and the instance was classified using Naive-Bayes trained on that specific cluster.The performance of the learner was tested on different number of clusters k ranging from 2 to 7
- Naive-Bayes with discretization (10 bins). InfoGain applied on the data
  After discretizing the data InfoGain was applied on the whole data set to retrieve n number of features (columns) where n is a user specified value ranging from 2 to 14.
- Filter the features with InfoGain. Apply Naive-Bayes with clustering
  In this experiment InfoGain was applied to select the n best features. After selecting the best features a new

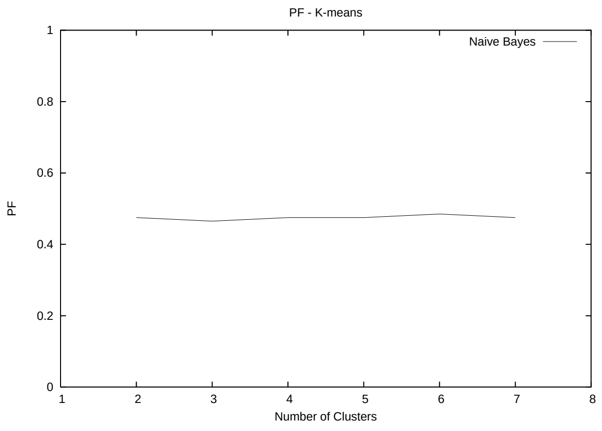
[1]http://promisedata.org/

dataset was build out of the original dataset containing only the selected features returned by InfoGain. The new dataset is clustered using K-Means. The learner is tested using different combinations of n-number of features and k-number of clusters

Running the first learner, Naive-Bayes without discretization on different datasets shows clearly that discretiztion effect the performance of Naive-Byaes. Figure 1 and 2 show the PD PF values after running the learner with different k number of clusters.



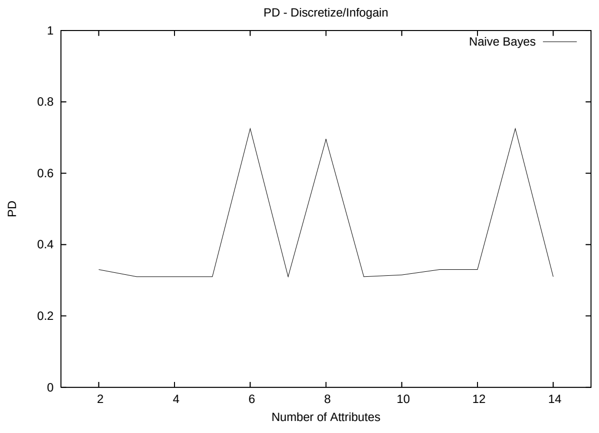
: Figure 1. PD: Naive-Bayes without discretization



: Figure 2. PF: Naive-Bayes without discretization

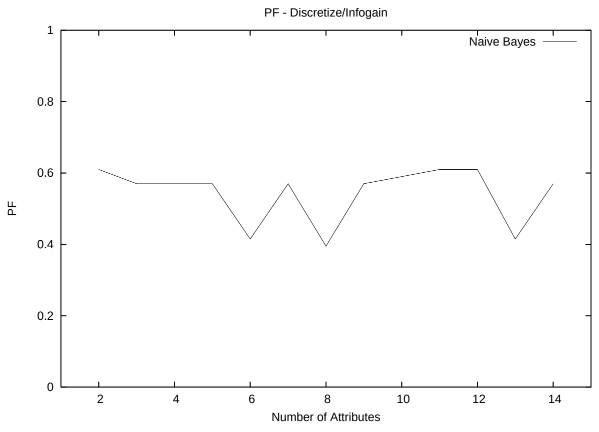
PD and PF are ploted agains the number of clusters k. Without discretization Naive-Bayes does not perform well. PD is no higher than 60 %. The number of clusters computed by k-means does not seem to influence the prediction probability however a relatively better PD are retrieved with 4, 5 and 7 clusters. This results also corresponds with [2] which state that discretization influence Naive-Bayes performance.

Applying InfoGain on discretize data outperforms the first learner. Data is discretized and InfoGain is used to extract the best n features (columns) from the data. Than Naive-Bayes is applyed on the closest cluster of each instance in the train set.Figure 3 and 4 show the PD and PF of this learner.



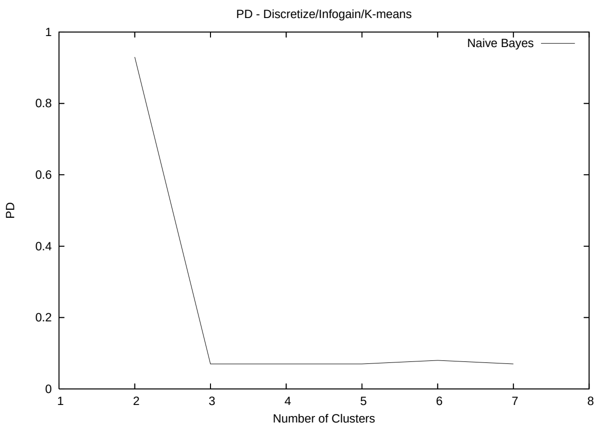
: Figure 3. PD: InfoGain and Naive-Bayes

The second learner scores the highest values for PD when the number of attributes selected by InfoGain n is 6 8 and 13. The max PD values are close to 80 %. In this case the advantage of discretizing the data and applying feature subset selection gives better results than applying the clustering on undiscretized data.

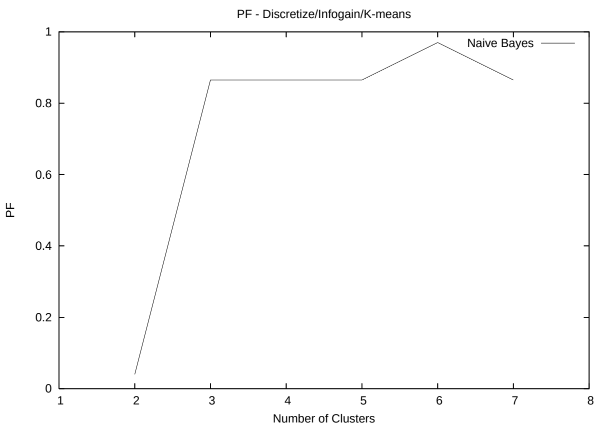


: Figure 4. PF InfoGain and Naive-Bayes

The third learner uses the information supplied by InfoGain to build a new data set containing only the features suggested by InfoGain. Discretization clustering is performed on the data using k-means and Naive-Bayes is trained on each cluster. This learner is exploring a combination of n-features and k-clusters in order to find the optimal solution. Figures 5 and 6 shows the PD and PF for the third learner.



: Figure 5. PD InfoGain/Clustering and Naive-Bayes



: Figure 6. PF InfoGain/Clustering and Naive-Bayes

The figures show PD and PF plotted against k-number of clusters which varies from 2 to 7. The PD values increase sharply when 3 or more clusters are used. The PD remain unchanged as the number of clusters increases. The PD values are relatively high (more than 90 %). These high results should be confirmed by performing further testing as to verify the validity of the learner.

Table 1: Statistics for each learner

Learner	PD	PF	Balance
1	0.57	0.47	0.42
2	0.86	0.09	0.21
3	0.5	0.5	0.41

Table 2: Top Features selected by InfoGain

Feat.	Shared$_PC1$	Shared$_CN1$	Shared$_MW$
1	LOComment	I	L
2	LOCCodeAndComment	Uniq-Op	Uniq Opnd
3	Uniq-Opnd	Uniq-Opnd	LOC
4	I	L	LOCommen
5	Uniq-Op	LOComment	LOCCodeAndCo
6	Total-Opnd	LOC	I
7	LOC	Total-Op	BranchCoun
8	V	Total-Opnd	V(G)
9	Total-Op	B	V
10	B	V	B
11	LOCode	IV(G)	EV(G)
12	BranchCount	D	LOCode
13	L	BranchCount	Total-Op

Table 1 . Shows the median PD and PF values for the three learners

Table 2. Shows the top 14 columns selected by InfoGain

5. RELATED WORK

This project is inspired by the Cross-project Defect Prediction work conducted by Thomas Zimmerman et. al. who claimed that software defect prediction works well whenever there is sufficient data to train any models and that in the case where data is insufficient, cross-project defect prediction suffices[1]. In their experiment, they ran 622 cross-project predictions for 12 real-world applications including Microsoft's Internet Explorer and Mozilla Foundation's Firefox web browser, and their results indicated that simply using models from projects in the same domain or with the same process does not lead to accurate predictions. With respect to the experiments they conducted, they learned that Firefox could predict defects in Internet Explorer but not vice versa and they succumbed to the conclusion that this is so because Firefox has more files than Internet Explorer has binaries and that the probability of a software with more data is more likely to predict defects in software with relatively less amount of data or modules.

Their study led to conclusions that some attributes were less significant than others which is somewhat obvious but also lead to questions such as why defect-prediction is not transivitve. As in, with regards to the experiments they conducted, File system predicted defects in Printing and Printing predicted Clustering but File system did not predict Clustering.

6. CONCLUSIONS

The study undertaken and discussed in this paper involves investigating a criteria for predicting default-prone modules in both small and large-scale software systems. From the experiments performed on each of the data sets we obtained from NASA and SoftLab, we learned that within-company software defect prediction works when the data accumulated for prior versions of the software in question is available. The results from our self-test experiments validates this conclusion. In other words, creating train and test data from the same data set allows us to successfully build models for software defect prediction. This is especially feasible in cases where the company (whose datasets are being used in the experiment) has been around for years, for example NASA, Microsoft or the Mozilla Foundation; or older versions of its software have already been in use for years. The data generated from such older versions can be analyzed, learned on and used to build defect prediction models in order to predict defects in newer versions of the software.

Some researchers indicate that collecting data from case studies and subjecting it to isolated analysis is not enough because statistics on its own does not provide scientific explanations; and that "we need compelling and sophisticated theories that have the power to explain the empirical observations"[3]. However, in our experiments, we ran 10 tests on 10 datasets, each time with a different number of nearest-neighbors and number of columns to be selected via infogain, and despite the fact that these experiments were conducted on randomly selected objects from the different data sets and resulted in inconsistencies in the accuracy of defect predictions, the average probability of detection was relatively high while the probability of false alarm was relatively low. Of course, these values were obtained after intensive analysis on the results from applying learners on our datasets which we pre-processed using a number of machine learning / data mining methods as mentioned in section 3.1.

7. FUTURE WORK

The experiments for this project were conducted using data obtained from NASA and SoftLab for this project simply because they are 2 complete different companies that operate and function in different ways. The National Aeronautics and Space Administration (NASA) is an agency of the United States government that undertakes the nation's projects related to space exploration and SoftLab on the other hand is a research laboratory of computer engineering in Turkey that conducts research on cost and effort estimation, defect estimation/prevention, value based software engineering and software process development and typically writes software controllers for home appliances. It is worthy to note that these 2 companies mentioned have no ties with each other.

We believe that for better results and much better success rates at predicting software defects, regression algorithms could be applied to the datasets to model the data with the least error. That is, apply the learners to the datasets a number of times and discard arguments to algorithms such as k-nearest neighbor and infogain that germinated less desirable results. Also, the learning algorithms used could be altered to analyze additional data instances added to the data already learned on and determine whether the new data would be benefitial to predicting defects or not. Furthermore, this work could be extended to exploit datasets that comprised of more than 2 classes (TRUE, FALSE) and with multiple class columns (at least 2). Also, for any cross-company software defect prediction to be done, datasets from 2 or more different sources but with with similar columns will be required. In the case of our study, the data from SoftLab contained more metrics than that from NASA. This rendered it difficult to build models by learning on the 2 datasets thus experiment on only within-company.

Nonetheless, a number of other rule-based learners could be used rather than *NaiveBayes* as was used for this project. These include, but are not limited to, the *PRISM* algorithm which aims at inducing modular classification rules directly from the training set; *OneR*, *TwoR*, *RIPPER*, and even *HyperPipes*. Due to the constraint on time, we were unable to experiment on such learners but the outcome of applying each of them on the same datasets we used would be of great value to software engineers / data miners. Finally, because all the data sets we used were complete, the alrgorithms used in our experiments didn't account for missing values but in future implementations, this will have to be addressed since ignoring this problem can introduce bias into the models being evaluated and lead to inaccurate data mining conclusions [6]. This can easily be done by either ignoring the entire record, filling in with a global constant (not recommended since most data mining algorithms will regard it as a normal value), fill in with the attribute's mean or median, fill in with the most likely value (using regression, decisino trees, most similar records, etc) or using other attributes to predict the value of the missing data.

8. THE *BODY* OF THE PAPER

Typically, the body of a paper is organized into a hierarchical structure, with numbered or unnumbered headings for sections, subsections, sub-subsections, and even smaller sections. The command `\section` that precedes this paragraph is part of such a hierarchy.[2] LaTeX handles the numbering and placement of these headings for you, when you use the appropriate heading commands around the titles of the headings. If you want a sub-subsection or smaller part to be unnumbered in your output, simply append an asterisk to the command name. Examples of both numbered and unnumbered headings will appear throughout the balance of this sample document.

Because the entire article is contained in the **document** environment, you can indicate the start of a new paragraph with a blank line in your input file; that is why this sentence forms a separate paragraph.

8.1 Type Changes and *Special* Characters

We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the

[2]This is the second footnote. It starts a series of three footnotes that add nothing informational, but just give an idea of how footnotes work and look. It is a wordy one, just so you see how a longish one plays out.

heading of this subsection will be in a sans serif[3] typeface, but that is handled by the document class file. Take care with the use of[4] the curly braces in typeface changes; they mark the beginning and end of the text that is to be in the different typeface.

You can use whatever symbols, accented characters, or non-English characters you need anywhere in your document; you can find a complete list of what is available in the *LaTeX User's Guide*[?].

8.2 Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

8.2.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin. . .\end` construction or with the short form `$. . .$`. You can use any of the symbols and structures, from α to ω, available in LaTeX[?]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\rightarrow \infty}x=0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

8.2.2 Display Equations

A numbered display equation – one set off by vertical space from the text and centered horizontally – is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\rightarrow \infty}x=0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty}x_i=\int_{0}^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

8.3 Citations

Citations to articles [?, ?, ?, ?], conference proceedings [?] or books [?, ?] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [?]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide*[?].

This article shows only the plainest form of the citation command, using `\cite`. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

[3]A third footnote, here. Let's make this a rather short one to see how it looks.

[4]A fourth, and last, footnote.

Table 3: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ^2_1	1 in 40,000	Unexplained usage

Figure 1: A sample black and white graphic (.eps format).

8.4 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material is found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

8.5 Figures

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of **.eps** and **.ps** files to be displayable with LaTeX. More details on each of these is found in the *Author's Guide*.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. and don't forget to end

Table 4: Some Typical Commands

Command	A Number	Comments
`\alignauthor`	100	Author alignment
`\numberofauthors`	200	Author enumeration
`\table`	300	For tables
`\table*`	400	For wider tables

Figure 3: A sample black and white graphic (.eps format) that needs to span two columns of text.

Figure 2: A sample black and white graphic (.eps format) that has been resized with the `epsfig` command.

Figure 4: A sample black and white graphic (.ps format) that has been resized with the `psfig` command.

the environment with figure*, not figure!

Note that either **.ps** or **.eps** formats are used; use the `\epsfig` or `\psfig` commands as appropriate for the different file types.

8.6 Theorem-like Constructs

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. There are two forms, one produced by the command `\newtheorem` and the other by the command `\newdef`; perhaps the clearest and easiest way to distinguish them is to compare the two in the output of this sample document:

This uses the **theorem** environment, created by the `\newtheorem` command:

THEOREM 1. *Let f be continuous on $[a,b]$. If G is an antiderivative for f on $[a,b]$, then*

$$\int_a^b f(t)dt = G(b) - G(a).$$

The other uses the **definition** environment, created by the `\newdef` command:

Definition 1. If z is irrational, then by e^z we mean the unique number which has logarithm z:

$$\log e^z = z$$

Two lists of constructs that use one of these forms is given in the *Author's Guidelines.*

There is one other similar construct environment, which is already set up for you; i.e. you must *not* use a `\newdef` command to create it: the **proof** environment. Here is a example of its use:

PROOF. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L.$$

Then

$$l = \lim_{x\to c} f(x) = \lim_{x\to c} \left[gx \cdot \frac{f(x)}{g(x)} \right] = \lim_{x\to c} g(x) \cdot \lim_{x\to c} \frac{f(x)}{g(x)} = 0 \cdot L = 0,$$

which contradicts our assumption that $l \neq 0$. $\square$

Complete rules about using these environments and using the two different creation commands are in the *Author's Guide*; please consult it for more detailed instructions. If you need to use another construct, not listed therein, which you want to have the same formatting as the Theorem or the Definition[**?**] shown above, use the `\newtheorem` or the `\newdef` command, respectively, to create it.

A *Caveat* for the TEX Expert

Because you have just been given permission to use the `\newdef` command to create a new form, you might think you can use TEX's `\def` to create a new command: *Please refrain from doing this!* Remember that your LATEX source code is primarily intended to create camera-ready copy, but may be converted to other forms – e.g. HTML. If you inadvertently omit some or all of the `\def`s recompilation will be, to say the least, problematic.

9. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

10. REFERENCES

[1] Cross-project defect prediction: A large scale experiment on data vs. domain vs. process.

[2] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. *Morgan Kaufmann*, 1995.

[3] N. E. Fenton and M. Neil. A critique of software defect prediction models. In *IEEE Transactions On Software Engineering*, pages 675–687, October 1999.

[4] E. Frank, M. Hall, and B. Pfahringer. Locally weighted naive bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.

[5] G. Gay, T. Menzies, and B. Cukic. How to build repeatable experiments. In *PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, pages 1–9, New York, NY, USA, 2009. ACM.

[6] J. F. K. Marvin L., Brown. Data mining and the impact of missing data. In *Industrial Management Data Systms*, pages 611–621. MCB UP Ltd, October 2003.

[7] B. Turhan, T. Menzies, A. Bener, and J. Distefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, November 2009.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX Expert

A.3 Conclusions

A.4 Related Work

A.5 Future Work

A.6 Acknowledgments

A.7 Additional Authors

This section is inserted by LATEX; you do not insert it. You just add the names and information in the `\additionalauthors` command at the start of the document.

A.8 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B. MORE HELP FOR THE HARDY

The sig-alternate.cls file itself is chock-full of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LATEX, you may find reading it useful but please remember not to change it.