Development of a Model for In-Silico Prediction of Carcinogenicity in Chemical Compounds

1. Introduction

Drug discovery and development is a lengthy, costly, and high-risk process. On average, it takes about 10-15 years for each new drug to be approved for clinical use, with an estimated cost of $1-2 billion dollars. Despite the implementation of many successful strategies, over 90% of clinical drug development fails [1]. Toxicity is a leading cause of failure of drug candidates in preclinical and phase I clinical trials. This is because, in addition to the potency and specificity of a drug candidate, other factors such as tissue exposure must be considered. It's imperative to establish the optimum dose of a drug required to cure disease-targeted tissues without causing substantial harm to other healthy tissues. Identifying carcinogenic compounds is an integral part of the toxicity screening process in drug development and often involves a two-year carcinogenicity assay conducted in laboratory settings, using rodents. This method, aside from being very time-consuming, is also highly expensive and requires tens to hundreds of animals per test.

Computational (In Silico) methods have become a popular alternative for various toxicological endpoints. These methods utilize computer-based models and algorithms to predict the carcinogenicity of chemicals or compounds without the need for traditional experimental testing. They leverage data on chemical structure, physicochemical properties, and biological activity to estimate potential toxicity. Given the challenges mentioned earlier, a prediction model such as this could be employed in the early stages of drug discovery. Hence, developers can assess their chances of success and explore optimization techniques long before a significant investment of time and resources has been made. Several attempts have been made in the development of In Silico models for carcinogenicity prediction. Li T et al. [2] have proposed DeepCarc - a deep learning-based (DL) model to predict carcinogenicity for small molecules using model-level representations. This classification model was developed using 692 unique chemical compounds and yielded a Matthews Correlation Coefficient (MCC) of 0.432, with an average improvement rate of 37%. In [3], Chen Z et al. have proposed DCAMP, a DL model based on capsule network and attention mechanism for molecular carcinogenicity prediction. This classification model was trained on 1564 chemical compounds using their molecular fingerprints and molecular graphs and achieved an average accuracy of 0.750 on an external validation dataset containing 100 compounds. In [4], Wang Y-W et al. have proposed CapsCarcino, another DL-based classification model to predict carcinogenicity. One notable uniqueness of CapsCarcino is its compatibility with small-sized training datasets. The model was constructed using the Dynamic Routing algorithm of the capsule network and achieved an accuracy of 0.85 on a validation dataset. In [5], Sarita Limbu and Sivanesan Dakshanamurthy utilized a Hybrid Neural Network (HNN) Deep Learning method to develop a carcinogenicity prediction model. The HNN model achieved an accuracy of 0.795 with a specificity of 67.3% for binary classification and was trained on 7994 different chemical compounds.

Beyond the traditional binary classification method commonly applied, carcinogenicity prediction has also been approached as a regression problem. For instance, in [5] and [6], regression models to predict the carcinogenic potency of chemical compounds were proposed. This quantitative approach aims to predict the concentration at which a mixture becomes carcinogenic to its target population.

Despite the impressive work already done in *in silico* model development, these models have yet to attain general regulatory acceptance, partly due to a lack of a generally accepted protocol for performing such operations and limitations in predictive performance and scope [7]. Therefore, concerted efforts are still required for the continued improvement of this method. This paper proposes a Gradient Boosting (GBoost)-based binary classification model for *in silico* prediction of carcinogenicity in chemical compounds. The model was trained on 5367 observational carcinogenicity assays involving [**SPECIFY]** unique chemical compounds.

2. Materials and Methods

About the Dataset

The dataset for this project was acquired from the PubChem online chemical repository [8]. It comprises observational laboratory results uploaded by independent contributors to PubChem. Each entry includes a unique identifier of a chemical substance and other experiment details. To enrich the data further for enhanced granularity, additional details about each chemical compound, such as its known toxicity property, were included.

Inclusion Criteria

The feature inclusion criteria closely followed Lipinski's Rule of Five (Ro5) as applied in traditional laboratory settings. In the drug discovery setting, the Rule of 5 predicts that poor absorption or permeation is more likely when there are more than 5 H-bond donors, 10 H-bond acceptors, the molecular weight is greater than 500, the calculated Log P (Clog P) is greater than 5, and a polar surface area of less than 140 Å^2 [9], hence, the dimensional data included for the training of this model encompassed these features. Since these drug-like properties have been known to influence drug outcomes, this study assumes the hypothesis that it is also possible to predict toxicity using them. Finally, measures were put in place to eliminate entries that were heavily correlated with each other. For instance, there is no need to include the IUPAC name of a compound when we already have its SMILES string as a unique identifier.

Data Cleaning and Pre-processing:

The data were aggregated and cleaned in Python [10] within the KNIME Analytics platform [11] to remove irrelevant features, standardize the values of some features, handle missing data, among other data cleaning operations. Afterwards, the cleaned and consolidated data were then passed to another Python Script node for data preprocessing, as follows:

Dimensionality reduction:

To reduce the dimensionality of the features, measures were implemented as follows: For each categorical feature in the original dataset, a threshold was set, and a value count was performed to determine the count of each class. All members of such feature with a count less than the threshold were replaced with a default value.

Morgan's fingerprint generation:

The Morgan fingerprint representation of the chemical compounds was generated using the RDKit library [12]. This fingerprinting technique captures important structural information about the molecule and works by using binary digits (0 or 1) to encode the presence or absence of substructures (e.g., functional groups, molecular fragments) within a molecule. To further reduce the dimensions of the dataset, features whose fingerprint was all-zeros were removed from the resulting data frame. The original SMILES strings were then replaced with the fingerprints.

Label Encoding:

To ensure that all data, including categorical data, were used in training the model, all categorical features were one-hot encoded into a numerical format that can be utilized by machine learning algorithms. This process converts categorical variables into binary vectors, representing the presence or absence of each category.

Handling missing data:

The Imputer library was utilized to fill missing data using the [METHOD: Mean, Median, etc.] method. This method replaces missing values with a statistical measure such as the mean or median of the feature across the dataset.

Checking for Class Imbalance:

The target feature was fairly balanced with an imbalance ratio of 1.25. This includes 3,726 experimental results that turned carcinogenic and 2983 that did not.

Feature Normalization:

The numerical features were normalized using the Z-Score normalization approach:

$$X_{\text{standardized}} = \frac{\sigma X - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of the feature.

3. Machine Learning Models

The Partitioning node randomly split the pre-processed data into training and testing sets with an 80:20 ratio. This step allows for training the model on a subset of the data and then evaluating its performance on a separate subset. A CSV Writer node connected to the second output port of the partitioning node dynamically saved the testing set to a local CSV file.

This step ensured a permanent separation of the training and testing sets, mitigating the risk of data leakage between them. The remaining 80% of the dataset was then utilized for training the models.

Four classification algorithms, namely Random Forest, Decision Tree, Gradient Boosting (GBoost), and Naïve Bayes classifiers, were trained simultaneously. Each classifier received a copy of the 80% split of the original dataset and underwent further splitting using a K-Partitioning node to create training and validation sets. The KNIME K-Partitioner node splits the input dataset into K partitions and sets up a loop for K iterations. During each iteration, fraction of the training set, of size $\left(\frac{k-1}{k}\right)S$, was used for training the model, while the remaining set of size $(1/k)S$ was used for testing the model, where k represents the number of cross-validations to be performed, and S denotes the sample size of the training dataset. The training set was connected to the classification learner node (learner), producing a trained model as output. This model was then passed to a predictor node along with the validation set. The cross-validation loop was terminated using a k-partitioning node terminator, and a scorer node was connected to evaluate the model's performance on the validation set. Additionally, each trained model was dynamically saved to a local file by connecting the output of the learner node to a model writer node. These saved models would be used to evaluate the performance of the models on the testing data set.

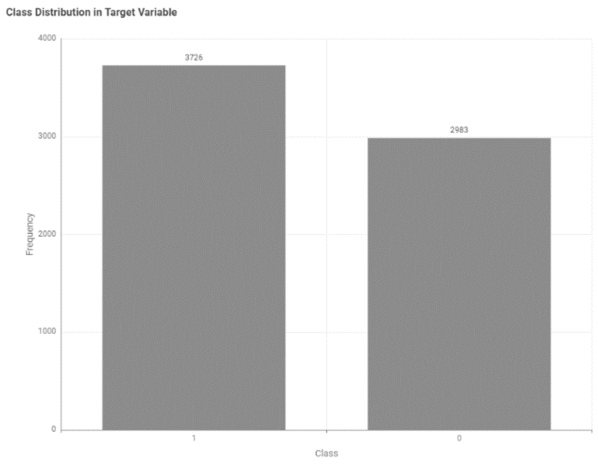


Figure 1: *Class Distribution within Target Feature*

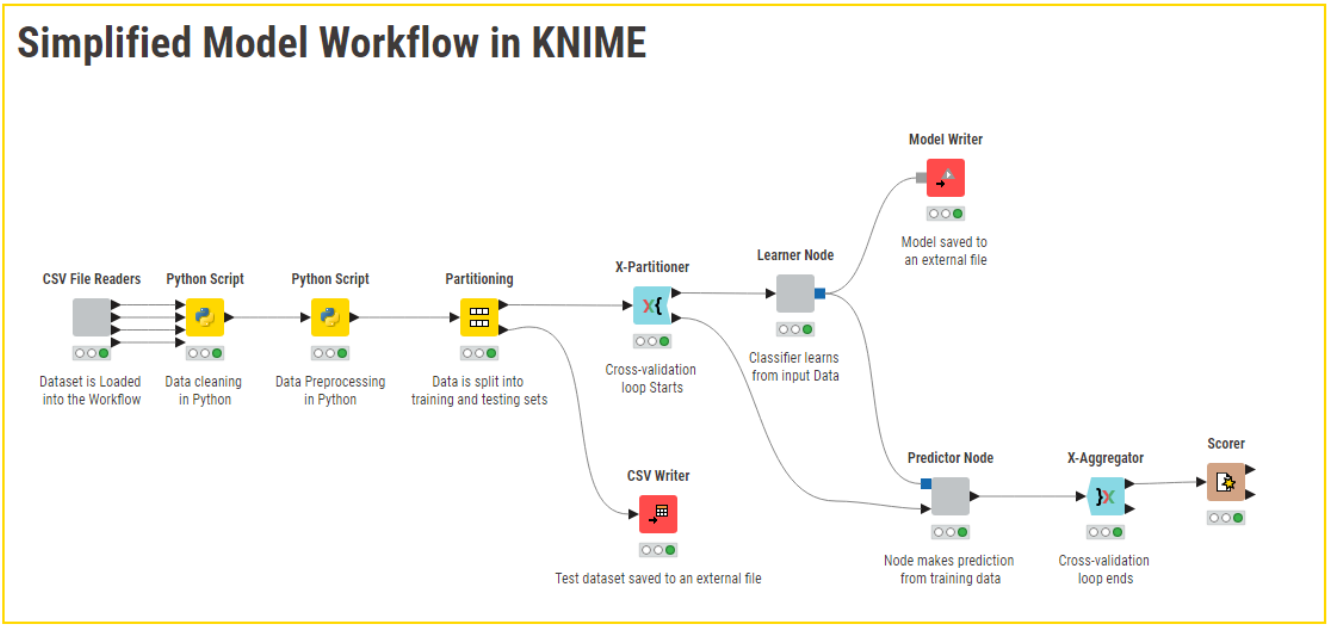
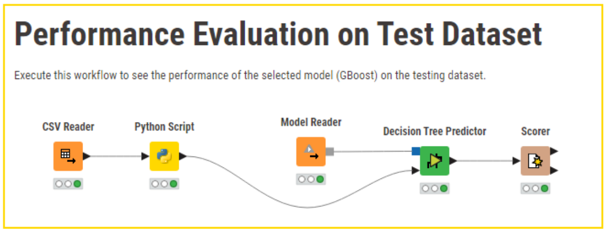

Simplified Model Workflow in KNIME
CSV File Readers
Dataset is Loaded into the Workflow
Python Script
Data cleaning in Python
Python Script
Data Preprocessing in Python
Partitioning
Data is split into training and testing sets
X-Partitioner
Cross-validation loop Starts
Learner Node
Classifier learns from input Data
Model Writer
Model saved to an external file
CSV Writer
Test dataset saved to an external file
Predictor Node
Node makes prediction from training data
X-Aggregator
Cross-validation loop ends
Scorer

Figure 2: Simplified View of the main workflow featuring nodes for data cleaning, preprocessing, consolidation and model development

4. Model Evaluation

Overview of the Model Evaluation Workflow

The workflow begins with a CSV Reader node, which loads the CSV file containing approximately 1,342 randomly drawn observations that were not used during model training. The output table from the CSV Reader node is then passed to a Python Script node for basic preprocessing. The models are then loaded into the workflow using Model Reader nodes in KNIME. The output of each model reader node, alongside the output of the Python script node, is then passed to the predictor node to generate predictions. Finally, a scorer node is connected to the output of each predictor node to display the performance results.


Performance Evaluation on Test Dataset

Execute this workflow to see the performance of the selected model (GBoost) on the testing dataset.

Figure 3: Workflow for Model Evaluation on Testing Dataset

5. Results and Discussion

The classification models were evaluated on the training dataset for predictive performance using accuracy, sensitivity, and specificity as metrics. Subsequently, the performance of the models was assessed on the testing data set using metrics such as accuracy, recall, precision, and the area under the receiver operating characteristics curve (AUC). The formulas for these metrics are provided below:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} X\ 100$$

$$Sensitivity = \frac{TP}{TP + FN} X\ 100$$

$$Specificity = \frac{TN}{TN + FP} X\ 100$$

$$Precision = \frac{TP}{TP + FP} X\ 100$$

$$Recall = \frac{TP}{TP + FN} X\ 100$$

where TP = True Positive, TN = True Negative, FP = False Positive and FN = False Negative.

As depicted in *Figure **6***, the Random Forest model achieved the highest accuracy at 77.70%, closely followed by the Decision Tree model. GBoost closely trailed in third place with an accuracy of 76.19%. However, Naïve Bayes performed poorly with an accuracy of only 55.54%. Similar trends were observed for the sensitivities of the models, except for Naïve Bayes, which achieved a sensitivity of 100%. The other three models followed closely, with GBoost at 81.58%, Decision Tree at 80.98%, and Random Forest at 80.48%, respectively. Regarding specificity, Naïve Bayes once again exhibited the poorest performance at 0%, while Random Forest led with a specificity of 74.22%. An analysis of the models reveals a close performance tie between GBoost, Decision Tree and Random Forest, with Naïve Bayes consistently displaying the weakest performance across most metrics.

Model	**Accuracy**	**Sensitivity**	**Specificity**
Naïve Bayes	55.54	100	0
GBoost	76.19	81.58	69.45
Decision Tree	77.31	80.98	72.72
Random Forest	77.70	80.48	74.22

Figure 4:Performance Statistics of the Models on Training Dataset

Figure 5 below shows the performance result of the models on the test dataset. Similar to the result on the training dataset, the Naïve Bayes model once again recorded the poorest accuracy and precision at 55.51% and 55.51% respectively. On the test dataset, the Decision Tree model achieved highest accuracy at 75.93%, closely followed by the Random Forest and GBoost models respectively. On the other hand, Random Forest achieved the highest precision at 78.53%. Naïve Bayes achieved the highest recall. In *Figure **6***, Random Forest achieved the highest performance with an AUC value of 0.844.

Model	**Accuracy**	**Precision**	**Recall**
Decision Tree	75.93	77.40	80
GBoost	75.11	75.53	81.61
Random Forest	75.78	78.53	77.58
Naïve Bayes	55.51	55.51	100

Figure 5: Performance Statistics of the Models on Testing Dataset

Naïve Bayes which showed high sensitivity (100%) on the training dataset, did not generalize well to the testing dataset where it exhibited the lowest performance. This could be due to the naive assumption of independence among features, which might not hold true in complex datasets. On the other hand, the Ensemble Methods (GBoost, Decision Tree, Random Forest) demonstrated relatively consistent performance between training and testing datasets, indicating good generalization ability. Decision Tree and Random Forest models showed slight drops in performance on the testing dataset compared to the training dataset, suggesting some degree of overfitting but still maintaining good overall performance. Random Forest generally performed slightly better than Decision Trees and GBoost in terms of accuracy and precision, especially on the testing dataset. GBoost, on the other hand, exhibited the highest recall on the testing dataset, suggesting it might be better at capturing true positives, albeit with slightly lower precision compared to Random Forest. Overall, although Random Forest demonstrated the most balanced performance among those tested, with relatively high accuracy, precision, and recall on the testing dataset, given that carcinogens portend a great threat to humanity and should be avoided at all costs, GBoost with the highest recall is proposed in this paper as it demonstrated the highest capability to capture all potential carcinogens, even at the expense of some false positives.

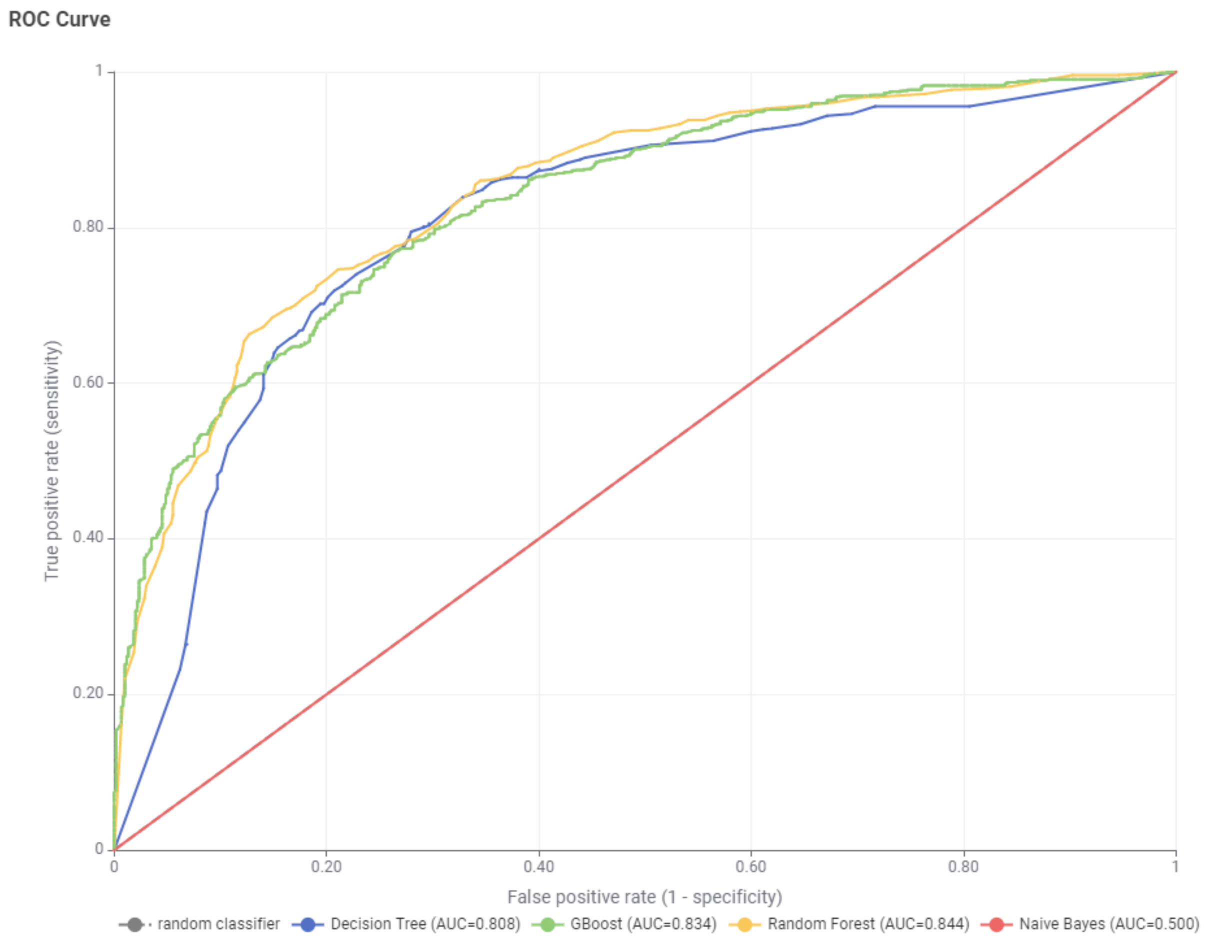


Figure 6: Receiver Operating Characteristics Curve (ROC)

6. References

1. Sun D, Gao W, Hu H, Zhou S. Why 90% of clinical drug development fails and how to improve it? Acta Pharm Sin B. 2022 Jul;12(7):3049-3062. doi: 10.1016/j.apsb.2022.02.002. Epub 2022 Feb 11. PMID: 35865092; PMCID: PMC9293739.
2. Li T, Tong W, Roberts R, Liu Z, Thakkar S. DeepCarc: Deep Learning-Powered Carcinogenicity Prediction Using Model-Level Representation. Front Artif Intell. 2021 Nov 18;4:757780. doi: 10.3389/frai.2021.757780. Erratum in: Front Artif Intell. 2022 Nov 28;5:1046668. PMID: 34870186; PMCID: PMC8636933.
3. Chen Z, Zhang L, Sun J, Meng R, Yin S, Zhao Q. DCAMCP: A deep learning model based on capsule network and attention mechanism for molecular carcinogenicity prediction. J Cell
4. Wang Y-W, Huang L, Jiang S-W, Li K, Zou J, Yang S-Y. CapsCarcino: A novel sparse data deep learning tool for predicting carcinogens [Internet]. Food and Chemical Toxicology, Volume 135, 2020, 110921, ISSN 0278-6915, https://doi.org/10.1016/j.fct.2019.110921
5. Limbu S, Dakshanamurthy S. Predicting chemical carcinogens using a hybrid neural network deep learning method [Internet]. Multidisciplinary Digital Publishing Institute; 2022 [cited 2024 Apr 9]. Available from: https://doi.org/10.3390/s22218185
6. Fradkin P, Young A, Atanackovic L, Frey B, Lee LJ, Wang B. A graph neural network approach for molecule carcinogenicity prediction. Bioinformatics. 2022;38(Supplement_1):i84-i91. doi:10.1093/bioinformatics/btac266.
7. Tice RR, Bassan A, Amberg A, Anger LT, Beal MA, Bellion P, Benigni R, Birmingham J, Brigo A, Bringezu F, Ceriani L, Crooks I, Cross K, Elespuru R, Faulkner DM, Fortin MC, Fowler P, Frericks M, Gerets HHJ, Jahnke GD, Jones DR, Kruhlak NL, Lo Piparo E, Lopez-Belmonte J, Luniwal A, Luu A, Madia F, Manganelli S, Manickam B, Mestres J, Mihalchik-Burhans AL, Neilson L, Pandiri A, Pavan M, Rider CV, Rooney JP, Trejo-Martin A, Watanabe-Sailor KH, White AT, Woolley D, Myatt GJ. In Silico Approaches In Carcinogenicity Hazard Assessment: Current Status and Future Needs. Comput Toxicol. 2021 Nov;20:100191. doi: 10.1016/j.comtox.2021.100191. Epub 2021 Sep 23. PMID: 35368437; PMCID: PMC8967183.
8. PubChem [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; 2004-. PubChem Bioassay Record for AID 1259411, CCRIS carcinogenicity studies, Source: Chemical Carcinogenesis Research Information System (CCRIS); [cited 2024 Apr. 9]. Available from: https://pubchem.ncbi.nlm.nih.gov/bioassay/1259411
9. Lipinski CA, Lombardo F, Dominy BW, Feeney PJ. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Adv Drug Deliv Rev. 2001 Mar 1;46(1-3):3-26. doi: 10.1016/s0169-409x(00)00129-0. PMID: 11259830.
10. Python Software Foundation. Python Language Reference, version 3.5. Available from: https://www.python.org/doc/versions
11. KNIME Analytics Platform. Available from: https://www.knime.com/knime-analytics-platform.
12. RDKit: Open-Source Cheminformatics Software. URL: https://www.rdkit.org.
13. Breiman L. Random Forests. Machine Learning. 2001; 45:5–32. https://doi.org/10.1023/A:1010933404324
14. Quinlan, J.R. Induction of Decision Trees. Machine Learning. 1986; 1:81–106. http://dx.doi.org/10.1007/BF00116251
15. Friedman J.H. Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics. 2001; 29:1189–1232. https://doi.org/10.1214/aos/1013203451
16. Rish I. An Empirical Study of the Naive Bayes Classifier. IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Seattle, 4 August 2001, 41-46.