

# Glossary

**Abstract syntax tree** The syntactic structure of source code represented as a graph (specifically, as a tree) in which vertices represent syntactic entities and edges represent the containment relationship. As syntax is language-specific, abstract syntax trees are also language-specific. A *concrete syntax tree* (or *parse tree*) is a similar representation constructed during the parsing of source code, e.g., by a compiler; it will contain redundant information and will fail to distinguish structures that differ due to semantic context. In software engineering, all such representations are typically referred to as abstract syntax trees, without this differentiation.

**Accuracy** [*Within information retrieval*] For binary classification of items, the percentage of the items available that are either correctly recommended or correctly not recommended. It can equivalently be interpreted as a probability, rather than a ratio. The measure is defined as

$$\frac{TP + TN}{TP + TN + FP + FN};$$

see confusion matrix for the definition of these quantities.

[*Within science and engineering generally*] The degree of closeness of measurements of a quantity to that quantity's actual value. Compare precision.

**Adaptability** A property of a system that indicates its ability to adapt automatically to changing conditions.

**Adaptivity** See adaptability.

**Anomaly detection** A data mining technique in which items are detected that do not conform to other patterns in the data, or in which patterns are detected that do not conform to expected patterns.

**Antecedent** See association rule.

**Application programming interface** The specification of how external agents should access functionality programmatically.

**API** See application programming interface.

**Argument** See validity [within mathematics].

**Association rule** A relation between two sets of items (*itemsets*), called the *antecedents* and *consequents*, in which the presence (equivalently, their truth, their relevance, etc.) of the antecedents implies the presence of the consequents.

**Association rule mining** Any mechanical process in which association rules are inferred within a dataset. Such a process operates atop a finite set of real data, in which transactions occur involving multiple items; if the same items frequently occur together, an association rule can be inferred. As the inferences of such a process may be false and may need to deal with anomalous cases, two concepts can be used in considering the quality of the mined rules: *support* and *confidence*. The support for an itemset is defined as the proportion of commits that contain the itemset. The confidence of an association rule is defined as the proportion of commits in which the antecedent is true for which the rule is correct, and hence the consequent is also true.

**AST** See abstract syntax tree.

**Attribute** A characteristic of a class of entities, whose specific value sometimes varies between instances of the class. In some circumstances, the instances themselves are described as possessing individual attributes.

**Availability** The ability of a user to obtain or access a system. This may be affected by factors such as system load, network connections, or the monetary cost of accessing the system.

**Balanced F-score** See F-measure.

**Benchmark** A standardized point of reference for a measurement.

**Benchmarking** The process of comparing something against standardized points of references in order to identify a best practice.

**Bias** The tendency to preferentially produce an outcome despite the existence of alternatives that are equally or more valid. This results in a *systematic error* in empirical data. Threats to validity can produce bias.

**Binary classification** See classification.

**Bug** See issue.

**Bug report** See issue report.

**Case study** A descriptive or explanatory analysis of an instance of a situation or phenomenon, useful as an exemplar of that instance and for the generation of hypotheses about other instances.

**Changeset** See commit.

**Classification** The assignment of items to two or more sets. Classification refers to both the process by which the assignment is created and the result. The special case of *binary classification* utilizes only two sets that are typically called true and false, with reference to whether the items in them are deemed relevant or not in some context. Comparison of actual classifications (arising from some form of real-world knowledge) and predicted classifications (arising from a mechanical interpretation of a model, called a *classifier*, such as a recommendation system) is an important means of assessing the quality of a classifier. Note that such an assessment may be invalid if the assumptions of the actual classifications are based on false premises. See confusion matrix.

**Classifier** See classification.

**Clickable link** A kind of navigation aid in which hypertext links are displayed, e.g., in an integrated development environment to allow the user to jump directly to corresponding code.

**Cluster analysis** The task of partitioning a set such that the items within a partition (called a *cluster*) are more similar in a defined sense than items in different partitions. It is an important method in exploratory data mining.

**Cold start problem** An issue arises when a recommendation system requires data to make its recommendations, e.g., extracted from a historical repository, but no such data yet exists.

**Collaborative filtering** A technique for generating recommendations in which the similarity of opinions of agents on a set of existing issues is used to predict the similarity of opinions of those agents on other issues. *Social tagging*, in which users label items as interesting, liked, recommended, etc., is one such technique.

**Commit** [*Within RSSEs*] A set of resources whose changes are added to a version control system together. For version control systems that do not support explicit commits of multiple resources, an inferred commit can be reconstructed where individually committed resources possess the same author and comment metadata, and the timestamps are in close proximity. Note that commit and *changeset* are synonymous under most situations, except where a distinction is needed between the resources changed contemporaneously and the resources added to a version control system together (e.g., when the author of the changeset differs from the author of the commit).

[*Within data management more generally*] A set of tentative changes that have been made permanent, typically at the end of a *transaction*.

**Confidence** See association rule mining.

**Configuration** An arrangement of parts and/or their parametrization by concrete values to obtain specific, well-defined instances that are interrelated in a well-defined manner.

**Confusion matrix** A means of categorizing the correctness of the mechanical classifications of a set of items. Confusion matrices are usually  $2 \times 2$  (for binary classification), though larger sizes and higher dimensionality are both possible. In both dimensions of the table are listed the possible classifications of the items under consideration; one dimension represents the actual or true classifications while the other represents the predicted or expected classifications from a classifier (e.g., a recommendation system). Each cell of the table records the number of items that have the corresponding combination of predicted and actual classifications. A perfect classifier will always agree with the actual classifications. In the special case of binary classification, the four cells of the table are given special names. *True positive* items *TP* are those that are correctly predicted by the classifier as true (equivalently, as yes, on, OK, of interest, etc.); *true negative* items *TN* are correctly predicted as false (equivalently, as no, off, not OK, not of interest, etc.). Cases where the classifier disagrees with reality are termed *false positives FP* (the classifier predicts true, but the actual classification is false) and *false negatives FN* (the classifier predicts false, but the actual classification is true). In many circumstances, the multiple values of

the confusion matrix are replaced by a smaller set of measures that combine the individual cell values in various ways; note that this replacement necessarily loses information. In statistics, false positives are known as *type I errors*, and false negatives as *type II errors*. See precision, recall, F-measure, accuracy, false negative rate.

**Consequent** See association rule.

**Content-based recommendation system** A recommendation system that makes recommendations based on items' content and profiles of users' interests.

**Context** The situation in which a system operates. The sense is usually limited to what is both visible and deemed relevant. Thus, a context could involve the physical location of a device, the characteristics of a task, the goals of a user, etc.

**Context awareness** A property describing a system that can sense its context and react accordingly.

**Corpus** A large collection of examples, systems, or other entities from which patterns can be inferred or recommendations can be drawn.

**Correctness** Usually, an imprecise term used in a situation where an item can be assessed as "right" or "wrong" (or sometimes as a fuzzy, intermediate value) to indicate the property of being "right." It implies that an independent means of this assessment is available that possesses greater validity if not absolute validity.

**Coverage** The percentage of items available to a recommendation system for which it is capable of making recommendations.

**Critiquing-based recommendation system** A knowledge-based recommendation system that supports a simple form of articulating preferences (e.g., higher performance, lower price, etc.).

**Cross-validation** A technique for model validation in which a set of pairs of queries and recommendations are repeatedly partitioned into training and test data, and the performance of the model is characterized over the different partitions. Variations exist for selecting the partitions, such as *k-fold cross-validation*, in which the dataset is partitioned into  $k$  equal sized subsamples; each subsample is then used as the validation data against which to evaluate the model that has been trained on the other  $k - 1$  subsamples. A related idea is *k-tail evaluation*, in which a sequence of data is partitioned into a suffix of  $k$  data items and a prefix of the remainder; the prefix is used as the training set and the suffix is used as the testing set. *k-tail evaluation* has less statistical validity, as the same data sequence would be repeatedly partitioned, leading to a lack of independence.

**Customization** The process of adaptation of an item for the sake of accommodating differences between individual contexts and/or users.

**Data cleaning** The process of detecting and correcting/removing data items that are corrupt or otherwise inaccurate. Also called *data cleansing* or *data scrubbing*.

**Data mining** The mechanical discovery of patterns within a (large) dataset. The term is often abused to mean any form of large-scale data processing. See association rule mining, cluster analysis, anomaly detection, machine learning.

- Density** A measure for characterizing datasets in which ratings have been made by users or inferred from users. It represents how many of the data items have been rated, as a percentage, per user, or overall, according to the circumstances.
- Developer** A human that interacts with the internal representations of software systems, such as source code or system designs. A developer acts as the user of various software tools, including integrated development environments and recommendation systems in software engineering. Developers are also known as *programmers* and *software engineers*. In our situation, we include *managers* who may know nothing about the programmatic entities within the software system.
- Developer context** The context in which the developer is working at a particular point in time, such as when a recommendation is generated.
- Developer profile** A set of attributes deemed useful for describing a developer, typically that will possess similarities to and differences from profiles of other developers.
- Diversity** A property of a set of recommendations wherein the recommendations are not considered trivial variations on each other.
- Domain analysis** The process of analyzing common terminology, problems, and solutions over a range of systems that share a common purpose.
- Edit location** Individual locations within an entity (typically the source code for a software system) that are modified by a developer to achieve some purpose. A high-level transformation (such as a refactoring) will be enacted as changes at individual edit locations, either automatically by a tool or manually by a developer.
- End-user** A human that uses a software system through a non-programmatic interface. The term is used to distinguish users who are not developers.
- Enhancement** See issue.
- Extensibility** A property of an item (typically, a programmatic entity) representing the ease with which it can be extended.
- Evaluation** An examination of a thing to assess its merits.
- Execution trace** A record of the execution of a program, typically listing the methods executed in the order in which they were executed. An execution trace may also record details of the objects and values passed, as well as metadata about the execution such as the time at which a method was entered and left and the thread in which the execution occurred. Statement-level execution traces are also common.
- Expected recommendation** The recommendation that ought to be obtained for a given input from a “perfect” RSSE. In many situations, the expected recommendation must be assumed based on an independent source of information (like real-world data).
- Experiment** A disciplined procedure to test a hypothesis, usually under (partially) controlled conditions. Or the act of following such a procedure.
- Explanation** A description of why an RSSE has chosen to produce a recommendation, generally presented to the user on demand and in context of that recommendation.

**Exploration** A collection of information about a previously unknown (physical or conceptual) space. Or the act of collecting such information.

**Failure** The externally visible evidence of a bug within a software system. A bug may never cause a failure if the bug is never executed. A failure may occur long after the execution of a bug.

**Fallout** The probability that a recommendation system will recommend a false item. It is equivalent to  $1 - \text{true positive rate}$ . See confusion matrix for a general overview.

**False negative** See confusion matrix.

**False negative rate** For binary classification of items, the percentage of the items predicted to be irrelevant that are actually relevant. It is used as a measure of quality of classifiers. It can equivalently be interpreted as a probability, rather than a ratio. The measure is defined as

$$\frac{FN}{FN + TN};$$

see confusion matrix for the definition of these quantities.

**False positive** See confusion matrix.

**False positive rate** For binary classification of items, the percentage of the irrelevant items that are predicted to be relevant. It is used as a measure of quality of classifiers. It can equivalently be interpreted as a probability, rather than a ratio. The measure is defined as

$$\frac{FP}{TN + FP};$$

see confusion matrix for the definition of these quantities.

**Feature** A distinguishing characteristic of an entity, intended to be positive in its target context.

**Feature request** See issue.

**Field study** A study conducted in a real-world setting, as opposed to an artificial one, such as a laboratory. A field study avoids control but merely observes phenomena, in the hope of minimizing influence on the phenomena and obtaining a richer set of observations.

**F-measure** A single measure that combines precision and recall. (Note that this necessarily loses information.) The *general F-measure*  $F_\beta$  is defined (for real, non-negative values of  $\beta$ ) as

$$(1 + \beta^2) \times \frac{\text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}.$$

More typically,  $\beta$  is set to the value of 1, producing the  $F_1$  *measure* (also called the *traditional F-measure* or *balanced F-score*); it is defined as

$$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

The  $F_1$  measure is typically called the F-measure for simplicity.

**$F_1$  measure** See F-measure.

**$F_\beta$  measure** See F-measure.

**Frequent itemset** See association rule mining.

**Functional requirement** A kind of requirement that focuses on the functionality aspects of a system, as opposed to the quality properties.

**Fuzzy set** A set of items for which a special function called a *membership function* is defined. The membership function maps each item to a value in the interval  $[0, 1]$  that represents the probability that that item is a member of the set.

**General F-measure** See F-measure.

**Generalizability** A property of an empirical result representing how well it would apply to situations other than those explicitly evaluated.

**Ground truth** Data collected from direct assessment as opposed to indirectly or remotely, i.e., “on the ground.” This matters as the validity of data collected through a series of inferences, or indirect interpretation is threatened at each step. On the other hand, ground truth data is often avoided due to high costs or risks associated with its collection.

**Group recommendation system** A recommendation system whose recommendations are aimed at a group as a whole, rather than individuals.

**Heuristic** A technique or value derived from experience, experimentation, or intuition from which a problem can be solved with no expectation of optimality. Heuristics are often used in situations where execution time is an important factor and suboptimal solutions are expected to suffice.

**Heuristics-based recommendation** An approach used in knowledge-based recommendation systems that uses heuristics in order to derive recommendations.

**Human–computer interaction** See user interface.

**Hybrid recommendation system** A recommendation system that combines two or more recommendation approaches in forming its recommendations, e.g., collaborative filtering, content-based, group, knowledge-based.

**IDE** See integrated development environment.

**Information retrieval** “Finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)” [30].

**Integrated development environment** A software application that provides a set of software tools for the development of other software. These software tools are typically integrated by sharing their internal representations of data (e.g., source code) and may also communicate directly with each other to reuse their functionality or to announce events to one another. Integrated development environments are typically intended to be extensible to new software tools. Integrated development environments aim to support the developer in their development tasks, focusing on usability for those tasks.

**Interaction data** Data collected from the users of a software system as they perform events with that system. Such data may be low level, recording keystrokes and mouse clicks at certain coordinates; or high level, recording presses of graphical button widgets, bringing a specific text editor to the foreground, or browsing for information. This data often consists of a description of the event that has occurred along with metadata such as the timestamp of the event. Inferring high-level interaction data from low-level interaction data is a nontrivial operation. *Navigation data* is a special form of interaction data in which the interactions (of interest) consist solely of moving between or within items, such as source code classes.

**Invalid** See validity [within mathematics].

**Issue** Constitutes two classes of entities: *bugs* and *enhancements*. A bug is a defect in a software system. An enhancement (or *feature request*) is a change that is desired to alter the system for reasons other than repairing a bug, such as extending functionality. These are grouped together under the generic label “issue” when it is not known or not important whether one is dealing with bugs or enhancements specifically. Bugs are also called *defects* and *problems*. Enhancements are also called *changes* and *problems*. Issues are also called *problems*. See issue report, issue repository, issue management system.

**Issue management system** A system that permits the recording of an *issue report*, as well as supporting the process of triaging, assigning, prioritizing, merging, and closing issue issues (i.e., the management of issues). An issue management system operates atop an issue repository.

**Issue report** Generally a structured report either describing a bug within a software system or requesting that a change be made; in other words, an issue report reports an issue. Issue reports are generally managed together, regardless of whether they constitute reports of bugs or enhancement requests. Issue reports typically collect metadata about the issue and about the management of the issue. See issue repository, issue management system.

**Issue repository** A collection of issue reports, stored in a specific manner, such as in a relational database. An issue repository is used by an issue management system to permit issues to be reported and managed through the process of addressing them.

**Issue triage** The lightweight analysis of a novel issue report to decide how to react to it, for example, to label it as a duplicate or of high priority.

**Itemset** See association rule.

**$k$ -fold cross-validation** See cross-validation.

**$k$ -furthest neighbors** A recommendation algorithm that recommends the  $k$  items that are least similar to a specified one (e.g., users that are dissimilar to the current user).

**$k$ -nearest neighbors** A recommendation algorithm that recommends the  $k$  items that are most similar to a specified one (e.g., users that are similar to the current user).

**$k$ -tail evaluation** See cross-validation.



**Knowledge capture** A process of explicitly recording in a tangible representation the knowledge possessed by a user.

**Knowledge-based recommendation system** A recommendation system that models knowledge about users and items in order to reason about which items meet a user's requirements. See critiquing-based recommendation system.

**Learner** See machine learning.

**Machine learning** A technique whereby a program (called the *machine learner* or simply the *learner*) can adapt according to the data it receives.

**Macroevaluation** A means of evaluating the quality of a recommendation system in which individual confusion matrices are populated with the results from individual recommendation trials. Each confusion matrix can then be summarized with standard measures, and measures of central tendency (such as the mean) can then be calculated over the individual measures.

**Manager** See developer.

**Metric** [*Within software engineering*] A measure of some specified property of entities within a defined set. Often, a given metric is intended to have greater meaning than its definition would automatically give it. A *validated metric* is thus a metric for which this greater meaning has been empirically validated to hold. For example, using a person's shoe size as a metric of intelligence would only be valid if we could demonstrate high correlation (or perfect correlation) between the two.

[*Within mathematics*] A function generalizing the notion of distance. A metric must conform to a certain set of properties: non-negativity, identity of indiscernibles, symmetry, and the triangle inequality.

**Microevaluation** A means of evaluating the quality of a recommendation system in which a confusion matrix is populated with the results from multiple recommendation trials without differentiating them. The confusion matrix can then be summarized with standard measures.

**Natural language processing** The automated interpretation of human language. This is more complex than the processing of programming languages due to a much greater presence of ambiguity and context-sensitivity in human languages.

**Navigation data** See interaction data.

**Network analysis** An analysis of a graph representing a set of entities and some relationship between them. This can be performed to characterize the overall shape of the network, to identify local properties, or to make decisions about the underlying entities or processes that the graph represents.

**Noise** Random data that does not carry information content but that can obscure the information content around it.

**Non-functional requirement** A kind of requirement that focuses on the quality aspects of a system, as opposed to its functionality.

**Novelty** The experience of discovering an item that is significantly different from others already known. Compare diversity and serendipity.

**Ontology** [*Within computer science*] The set of concepts that exist within a domain, and the relationships between those concepts. Note that a taxonomy is an ontology restricted to only include the subsumption relation (i.e., parent/child).

**Overfitting** Use of a statistical model to describe noise in a dataset rather than the relationship obscured by the noise. This can occur when the number of parameters in a model is close to the number of datapoints being fit, or when there has been no differentiation between the data used to derive the model and the data used to validate the model.

**Personalization** The delivery of different information (i.e., recommendations) depending on the target user.

**Persuasiveness** The capability of a recommendation system to influence a user's attitude, decisions, or behavior.

**Positive predictive value** See precision.

**Precision** [*Within information retrieval*] For binary classification of items, the percentage of the items predicted to be relevant that are actually relevant. It is used as a measure of quality of classifiers. It can equivalently be interpreted as a probability, rather than a ratio. The measure is defined as

$$\frac{TP}{TP + FP};$$

see confusion matrix for the definition of these quantities. A synonymous term used in other areas is *positive predictive value*. When the set of items predicted to be relevant is restricted to those above some threshold  $n$  (such as above some value of similarity) or the size of this set is constrained to  $n$ , we can speak of *precision at  $n$* .

[*Within science and engineering generally*] The degree to which repeated measurements of the same quantity under unchanged conditions agree. Compare accuracy.

**Prediction** A statement about the state of some entity derived only in part from the information possessed about it. Predictions often focus on the future state of an entity based on its current state and some model of change. Recommendations implicitly or explicitly predict the utility of the recommended item/action to the user.

**Privacy** The ability of an individual or group to selectively reveal information about themselves, when and if they so choose.

**Proactive recommendation** A recommendation that is presented to the user when the recommendation system deems it appropriate, without waiting for the user to request it.

**Program transformation** Any alteration or act of alteration of a program, usually conceived at the level of source code, but that could operate at higher or lower levels of abstraction.

**Programmer** See developer.

**Quality** An imprecise term denoting the fitness for purpose of a product or process. It may involve both objective and subjective elements, resulting in significantly different opinions of quality from different stakeholders.

**Reactive recommendation** A recommendation that is presented to the user only when the user requests it.

**Reactivity** The ability of a recommendation system to provide good quality recommendations in real-time according to some specified time threshold criterion.

**Recall** For binary classification of items, the percentage of the items that are actually relevant that are predicted to be relevant. It is used as a measure of quality of classifiers. It can equivalently be interpreted as a probability, rather than a ratio. The measure is defined as

$$\frac{TP}{TP + FN};$$

see confusion matrix for the definition of these quantities. Synonymous terms used in other areas are *true positive rate* and *sensitivity*. When the set of items predicted to be relevant is restricted to those above some threshold  $n$  (such as above some value of similarity) or the size of this set is constrained to  $n$ , we can speak of *recall at  $n$* .

**Recommendation** An information item estimated to be valuable in a given context. When the “estimate” is universally accurate, the information item is not a recommendation, but the correct answer.

**Recommendation box** The area wherein recommendations are displayed on an online surface.

**Recommendation system** A software application that provides information items estimated to be valuable for a task in a given context, i.e., recommendations. If the “estimate” is universally accurate, the system is not a recommendation system, but a system for computing the correct answer.

**Recommendation system in software engineering** A software application that provides information item estimated to be valuable for a software engineering task in a given context. If the “estimate” is universally accurate, the system is not an RSSE, but a system for computing the correct answer.

**Refactoring** Restructuring software to alter its internal structure without altering its external behavior. Such changes are typically performed in order to improve the internal properties of the software (such as its understandability or extensibility) without breaking external software agents or making end-users aware of the changes. Refactoring is both the general notion of such changes and specific transformations, especially when standardized (e.g., a *rename refactoring*).

**Reinforcement** [*Within RSSEs*] A heuristic measure defined by the Suade tool [10] for the likelihood of the relevance of an entity given its relationship with other elements, some of which are known to be relevant. According to the intuition of reinforcement, structural neighbors that are part of a cluster that contains many elements already in the set of interest are more likely to be interesting because they are the “odd ones out” in the cluster of elements related to the set of interest.

**Relevance** The value of an item to a specific user in completing a specific task at a specific time.

**Reporter** The user who has reported an issue.

**Representativeness** A property of a data item or sample that allows it to stand in for other items or the general population of interest. Representativeness can only be defined relative to a specific property or set of properties of interest. In mathematical terms, a sample could be considered representative if it is an element of an equivalence class under a pertinent equivalence relation. True representativeness is often difficult to assess when the population characteristics are not known.

**Reproducibility** The ability of an evaluation to be repeated in order to arrive at the same conclusions. The term is often meant more narrowly as the ability for an experiment to be repeated by different researchers to arrive at the same results. An irreproducible evaluation is generally not seen as valuable due to the possibility that it was conducted incorrectly and thus that the conclusions are not supported.

**Requirement** A condition or capability that must be met by a software product or software development process.

**Requirements elicitation/negotiation** A collaborative process, involving multiple stakeholders, of identifying requirements. As stakeholders' opinions may conflict as to the importance or value of individual requirements, negotiation is used to resolve conflicts.

**Response time** The time taken by a system to react to an input.

**Robustness** The ability of a system to cope with faults and failures.

**Root-mean-squared error** A measure comparing the values predicted by a model (i.e., a recommendation system) and the values actually observed. It is defined as

$$\sqrt{\frac{\sum_{i=1}^n (x_i - \hat{x}_i)^2}{n}},$$

where each  $x_i$  is the predicted value and each  $\hat{x}_i$  is the actual value.

**RSSE** See recommendation system in software engineering.

**Satisfiability** [*Within mathematics*] The problem of determining if there exists an interpretation that satisfies a logical (Boolean) expression.

**Scalability** The ability of something to accommodate growth reasonably, or to be adapted in order to accommodate growth reasonably. For example, this can mean that the growth in execution time does not exceed some bound relative to the input size. In principle, scalability should also support shrinkage at reasonable reduction in resource usage, but for many contexts, it is only the growth characteristic that is deemed important.

**SCoReS** See source code-based recommendation system.

**Sensitivity** See recall.

**Serendipity** The experience of finding an unexpected and fortuitous item. Compare novelty.

**Simulation** An imitation of the behavior of some process, usually for the purpose of studying that process. Equivalently, an imitation of the functioning of one system by another, usually simpler one.

**Social network** A graph consisting of actors (the vertices) and their relationships (the edges) in which the actors have a social existence (i.e., they will generally be humans) and their represented relationships will have a social significance.

**Social tagging** See collaborative filtering.

**Software configuration management system** See version control system.

**Software engineer** See developer.

**Software product line** A set of software systems that share a common, managed set of features, as opposed to possessing copies of those features.

**Software quality metrics** Any set of metrics used to measure the quality aspects of software products, projects, and/or processes.

**Sound** See validity [within mathematics].

**Soundness** See validity [within mathematics].

**Source code-based recommendation system** A recommendation system that produces recommendations principally by analyzing the source code of a software system.

**Specificity** [*Within RSSEs*] A heuristic measure defined by the Suade tool [10] for the likelihood of the relevance of an entity given its relationship with other elements, some of which are known to be relevant. According to the intuition of specificity, structural neighbors that have few structural dependencies are more likely to be interesting because their relation to an element of interest is more unique.

[*Within information retrieval*] See true negative rate.

**Speculative analysis** A dynamic analysis technique in which a range of possible actions are automatically tried, and the estimated quality of the results is used to rank the possibilities.

**Stakeholder** An entity (typically a person, but sometimes a group or organization) with an interest in the process or outcome of a project.

**Support** See association rule mining.

**Systematic error** See bias.

**Taxonomy** A classification of concepts or entities within a domain, and their parent/child relationships. See ontology.

**Test-driven development** A software development process in which an automated test case is written prior to the functionality that that test case is intended to exercise. The idea has been promoted in various agile software development methodologies for its potential to define the conditions for completion, and to avoid writing test cases that immediately pass rather than checking for correct behavior, which can happen due to the phenomenon of “debugging blindness.”

**Text link** A kind of navigation aid consisting of textual references, such as corresponding files and line numbers, e.g., see Chap. 5.

**Threat to validity** See validity [within evaluation].

**Traditional F-measure** See F-measure.

**Transaction** See commit.

**Transparency** A property of a recommendation that permits the user to understand why the recommendation has been made.

**Triangulation** [*Within evaluation*] Conducting multiple evaluations, typically via different methods and/or on different data sources, in order to improve the generalizability of the findings. Since each method and data source will have its own threats to validity, the intent is that the different methods/data sources will differ in their threats to validity, and thus, some threats can be shown to exist or not exist.

**True negative** See confusion matrix.

**True negative rate** For binary classification of items, the percentage of the items that are actually not relevant that are predicted to be not relevant. It is used as a measure of quality of classifiers. It can equivalently be interpreted as a probability, rather than a ratio. The measure is defined as

$$\frac{TN}{TN + FP};$$

see confusion matrix for the definition of these quantities. A synonymous term from other areas is *specificity*. When the set of items predicted to be relevant is restricted to those above some threshold  $n$  (such as above some value of similarity) or the size of this set is constrained to  $n$ , we can speak of *true negative rate at  $n$* .

**True positive** See confusion matrix.

**True positive rate** See recall.

**Trust** Reliance on the actions of an entity, such as a recommendation system. Trust can be established transitively through a recommendation by trusted entity, or directly through repeated observation of reliable actions. Trust can be lost by observation of unreliable actions. Believable explanations of behavior can help to establish trust.

**Type I error** See confusion matrix.

**Type II error** See confusion matrix.

**Understandability** A property that assesses a user's ability to correctly interpret the meaning of an item. It necessarily depends on the knowledge, experience, and skills of the user.

**Usability** A property that assesses a user's ability to easily use an item. This generally includes aspects of understandability.

**User** An agent external to a software system that makes use of that system. A user is usually a human being in the software engineering context, but in some other areas, software agents are also deemed to be users or other software entities. In the context of recommendation systems in software engineering, the user is usually a developer; for example, an RSSE residing within an integrated development environment would target a developer. The generic term *user* is taken to include both developers and end-users.

**User history** See interaction data.

**User interface** The portion of a software system that supports interaction with a user.

**User satisfaction** The extent to which a user is supported in their task by a recommendation.

**Valid** See validity [within mathematics].

**Validity** [*Within evaluation*] The extent to which the findings of an evaluation are well founded and correspond to reality. Every empirical method and every data source possesses one or more properties (called threats to validity) that may render findings derived therefrom to lack validity.

[*Within mathematics*] A property of a logical argument. A *valid argument* is one in which the truth of the premises necessitates the truth of the consequences, regardless of whether the premises are actually true; an argument that lacks validity is called *invalid*. This contrasts with a *sound argument*, which is one that is also valid, but whose premises are known to be true. *Soundness* is the analogous property that deals with the question of whether an argument is sound or not sound.

**Variant** See version.

**VCS** See version control system.

**Version** Given an entity (such as an entire software system), a version of that entity constitutes a particular set of changes to it from its original form. Different versions may coexist to support different purposes, or they may sequentially supplant older versions, or both. Synonymous terms are *revision* and *variant*. Specifically coexistent versions are also called *variants*.

**Version control system** A software system used to record incremental changes to resources, along with metadata describing those changes, such as the author, timestamp when the change was added, and a comment made by the developer who added the change. Version control systems can be subsumed more generally by the term *software configuration management systems*.

**Wizard of Oz experiment** An experiment in which apparently automated responses/recommendations are being faked, either hard-coded in the software being used to mediate the experiment or manually entered by an experimenter (usually in secret) during the experiment. (The name derives from a fictional character in American literature who was pretending to have great magical powers but was actually operating special effects from a hidden location.)

# Index

Note: Italicized page references point to (explicit or implicit) definitions.

## A

Abstract syntax tree (AST) 539  
A/B testing 238  
Accuracy 154, 250, 276, 278, 280–287, 292, 294, 295, 297, 439, 470, 539  
Active user 95  
AdaBoost 57  
Adaptability 200, 278, 290, 291, 539  
Adaptivity 215  
Affordance 229, 229  
Affordance overlay 231  
Annotation 229–231  
Anomaly detection 539  
ANOVA 52  
Antecedent. *See* Association rule  
Antipatterns 388  
Application programming interface (API) 3, 82–84, 94–97, 100, 104, 105, 107, 108, 111, 113, 115, 199, 252, 314, 316, 331, 539  
Application programming interfaces 94  
Apriori 40, 41, 41–43, 67  
Area under curve 282  
Argument. *See* Validity  
Assessability 227, 227  
Association rule 41, 97, 457, 540  
Association rule mining 97, 121, 122, 540  
Assumptions 323, 325  
AST. *See* Abstract syntax tree  
Attention-sensitive alerting 229  
Attribute 540

Availability 117, 540  
Average hit ratio 261  
Average rank 261  
Awareness 85

## B

Balanced F-score. *See* F-measure  
Benchmark 86, 116, 246, 268, 540  
Benchmarking 276–297, 540  
Bias 132, 247, 308, 319, 540  
Big Data 40, 41, 49  
Bigram 486  
Binary classification. *See* Classification  
Binary decision diagram 514  
Blockbuster 284  
BM25F 486  
Branching 312  
Bug 78, 84, 131, 254, 303. *See* Issue  
Bug prediction model 151  
Bug report 131, 132. *See also* Issue report

## C

C4.5 44, 44–47  
Canopy clustering 70  
Capturing context 6  
CART. *See* Classification and regression tree (CART)



- Case study 116, 239, 254, 540
  - Catalog coverage 253, 253, 283
  - Changeset. *See* Commit
  - Churn 4
  - Classification 152, 250, 482, 540
  - Classification and regression tree 46–49
  - Classifier 256. *See also* Classification
  - Click-through rate 288
  - Clickable link 110, 541
  - CLIFF 61
  - Clippy 224, 370
  - Cloud 3
  - Cluster analysis 114, 541
  - Code completion 90, 94
  - Code Conjurer 363, 372, 377
  - Code example 6
  - Code scavenging 369
  - Cognitive effort 225
  - Cognitive walkthrough 238
  - Cold start problem 30, 79, 79, 254, 283, 291, 469, 541
  - Collaborative filtering 7, 541, 16–541
  - Command 6
  - Command line 106
  - Commit 3, 79, 137, 312–314, 494, 541
  - Communication archive 3
  - Concrete syntax tree 539
  - Confidence 79, 97, 283, 286, 310, 312, 313, 471. *See* Association rule mining
  - Configuration 247, 304, 313, 541
  - Conflict set 26
  - Confusion matrix 251, 256, 310, 541
  - Consequent. *See* Association rule
  - Construct validity 310
  - Content glut effect 278
  - Content-based filtering 20
  - Content-based recommendation system 16, 287, 460, 478, 542
  - Context 4, 181, 193, 204, 247, 542
  - Context awareness 542
  - Control-flow analysis 311
  - Conversion rate 288
  - Coordination 85
  - Coordination requirements 214
  - Corpus 82, 88, 89, 94, 542
  - Correctness 132, 246, 248, 265, 542
  - Correlate 152
  - Cosine dissimilarity 67
  - Cosine similarity 484, 485, 489
  - Coverage 253, 265, 283, 284, 439, 472, 542. *See also* Catalog coverage, Prediction coverage
  - Critiquing-based recommendation 32
  - Critiquing-based recommendation system 32, 542
  - Cross-validation 154, 462, 542
  - Crosscutting concerns 424
  - Customer receiver operating characteristic curve 282
  - Customization 200, 542
- D**
- Dashboard 233
  - Data cleaning 114, 542
  - Data clustering 69–72
  - Data constraints 290
  - Data mining 77, 78, 81, 82, 84, 85, 90, 132, 312, 542. *See also* Supervised learning, Unsupervised learning
  - Data mining algorithms 39
  - Data mining for sparse tables 40. *See also* *k*-nearest neighbors
  - Data mining recommendation system 78
  - Data preprocessing 6
  - Data pruning 61–64
  - Data transformation. *See* Latent semantic indexing, Principal component analysis, Pruning, Support vector machine
  - Dataflow analysis 311
  - DBScan 69
  - Debugging 331
  - Decision tree learning. *See* C4.5, CART
  - Degree of interest 175, 179
  - Density 281, 543
  - Dependency 78, 84, 85, 113, 304, 311, 316, 317, 551
  - Dependency graph 78, 317
  - Deployment 84
  - Descriptive statistics 306
  - Design scavenging 369
  - Developer 93–96, 98, 100–108, 110–112, 114–117, 121, 123–126, 131, 223, 245, 254, 543
    - context 78, 81, 85, 90, 96, 362, 543
    - profile 201, 543
  - Developer-centric 315
  - Developer-specific analysis engine 84, 89
  - Development environment 3
  - Diagnosis 26
  - Dice’s coefficient 485
  - Differential privacy 264
  - Difficulty 256

Discounted cumulative gain 250  
 Discretization 43  
 Distraction 229  
 Diversity 246, 265, 284, 543  
 Domain 314  
 Domain analysis 465, 543  
 Domain-specific language 106  
 DOPLER 515  
 Driver 304, 324  
 Duplicate report 480

**E**

Edit location 424, 543  
 Elementary transformation 425  
 EM. *See* Expectation maximization  
 Email notification 233  
 Emergent behavior 306  
 End-user 103, 131, 247, 278, 543  
 Enhancement 135. *See also* issue  
 Ensemble learning 50, 256  
 Environment 132  
 Ericsson 480  
 Error-proneness 89  
 Evaluation 87, 88, 161, 238, 246, 276–297, 302, 301–325, 543  
 Example 94, 98, 107, 110, 111, 113, 115, 315  
 Example-based program transformation 435, 439  
 Example recommendation 362  
 Execution trace 3, 108, 543  
 Expectation maximization 40  
 Expected recommendation 307–310, 313–316, 318, 320, 321, 543  
 Experience atoms 211  
 Experiment 116, 239, 268, 302, 318, 543  
 Explanation 7, 104, 256, 543  
 Exploration 313, 325, 544  
 Extensibility 543

**F**

$F_\beta$  measure 545  
 F-measure 161, 251, 310, 464, 544  
 $F_1$  measure 544, 545  
 Failure 136, 544  
 Fallout 287, 544  
 False negative 166, 310. *See also* Confusion matrix

False negative rate 544  
 False positive 88, 114, 166, 227, 310, 448, 506. *See also* Confusion matrix  
 False positive rate 250, 544  
 FASTMAP 51  
 Fault localization 331  
 Fayyad–Irani 51, 55  
 Feature 247, 466, 544  
 Feature request 132. *See also* Issue  
 Field study 116, 239, 329–352, 544  
 File authorship matrix 214  
 File dependency matrix 214  
 Finding a needle in a haystack 319  
 Fishtail 227  
 Fitness for purpose 366  
 Framework 94, 96, 97, 104  
 Frequent itemset mining 189  
 Frequent itemsets 43, 114. *See also* Association rule mining  
 Functional requirement 285, 545  
 Fuzzy set 310, 545

**G**

Garbage-in/garbage-out 303  
 General F-measure 251. *See also* F-measure  
 Generalizability 90, 239, 305, 315, 321, 545  
 Generic change analysis engine 84  
 Gold standard 116, 248  
 Granularity level 182  
 Ground truth 308, 545  
 Group recommendation 32  
 Group recommendation system 32, 545

**H**

Hedges's test 43  
 Heuristic 78–91, 98, 100, 108, 114, 121, 141, 255, 312, 314, 315, 317, 545  
 Heuristic evaluation 238  
 Heuristics-based recommendation 513, 545  
 Hidden Markov model 189  
 Hint 96  
 History 2, 79, 95, 98, 107, 108, 118, 303, 304, 309, 312–314, 320, 321  
 Hitting set 26  
 Human–computer interaction 106. *See also* User interface  
 Hybrid recommendation 27  
 Hybrid recommendation system 27, 257, 545  
 Hypothesis 302

**I**

Icon 231  
 IDE. *See* Integrated development environment  
 Incremental learning 58–59  
 InfoGain 51, 66  
 Information overload 4, 278, 286, 331  
 Information retrieval 482, 545  
 Information space 2. *See also* Application programming interface, Communication archive, Development environment, Execution trace, History, Interaction trace, Source code  
 Integrated development environment (IDE) 82, 83, 94, 98, 105, 106, 120, 123, 126, 150, 174, 177, 200, 224, 256, 303, 312, 319, 320, 442, 493, 545  
 Intentions 181  
 Interaction data 174, 178, 205, 318–320, 322, 546  
 Interaction trace 3  
 Internet marketplace 278  
 Internet-scale software search engine 360  
 Intra-list similarity 255  
 Intrusiveness 106  
 Intuition 85, 86, 89, 91  
 Invalid 134. *See also* Validity  
 Issue 546  
 Issue management system 3, 108, 134, 206, 478, 546  
 Issue report 6, 84, 93, 131, 477, 546  
 Issue repository 132, 205, 477, 546  
 Issue triage 479, 546  
 Item. *See* Recommendation item  
 Item-based collaborative filtering 19  
 Item-space coverage. *See* Catalog coverage  
 Itemset 255. *See also* Association rule

**J**

Jaccard similarity coefficient 485

**K**

$k$ -fold cross-validation 309, 546  
 $k$ -furthest neighbors 294, 546

K-means 40, 70  
 $k$ -nearest neighbors 40, 281, 294, 459, 546  
 $k$ -tail evaluation 309, 320, 546  
 $k$ -tail multi-session 320  
 $k$ -tail multi-use 320  
 Kendall's  $\tau$  250  
 Knowledge capture 547  
 Knowledge-based recommendation 21  
 Knowledge-based recommendation system 16, 547

**L**

Latent semantic analysis 102  
 Latent semantic indexing (LSI) 62, 495.  
*See also* Latent semantic analysis  
 Learner. *See* Machine learning  
 Learning rate 261, 266, 283  
 Level of granularity 313  
 Lexical similarity. *See* Textual similarity  
 Library 94, 98, 104  
 Line 10 rule 206  
 Linear television 277, 291  
 LSI. *See* Latent semantic indexing

**M**

Machine learning 85, 89–91, 114, 152, 188, 258, 281, 547  
 Machine learning, important services of 52  
 Macroevaluation 249, 310, 547  
 Main trunk 312  
 Maintainability 89  
 Make-or-reuse dilemma 373  
 Manager. *See* Developer  
 Mapping bug reports 141  
 Master report 480  
 Mean absolute error 248, 282  
 Mean average precision 282  
 Meaningfulness 303, 305, 309, 315, 318  
 Medoid 468  
 Merge 312, 314  
 Meta-learner 57  
 Meta-result 305  
 Method 331  
 Methodology 331  
 Metric 113, 132, 246, 547  
 Microevaluation 249, 310, 547  
 Mini-batch K-means 70

Minkowski distance 67  
 Misclassification 140  
 Mockup 238, 305  
 Model-based diagnosis 25  
 Multi-attribute utility theory 24, 515  
 Multi-objective evaluation 276

**N**

Naive Bayesian 40, 52  
 Naming bug 84  
 Natural language processing 281, 547  
 Natural language queries 110  
 Navigation data. *See* Interaction data  
 Navigation patterns 175  
 NBTrees 71  
 Netflix Prize 276, 278, 281  
 Network analysis 153, 503, 547  
*n*-gram 486, 487  
 Noise 114, 132, 547  
 Non-functional requirement 285, 290, 547  
 Normalized discounted cumulative gain 250  
 Normalized distance-based performance measure 250  
 Normalized Google distance 288  
 Novel 257  
 Novelty 265, 284, 288, 547

**O**

Obviousness 225, 258  
 Offline data mining 88–89  
 Online data mining 88  
 Ontology 183, 547  
 Open-source system 314  
 Oracle 116, 361  
 Orthogonal variability model 515, 516  
 Overall learning rate 283  
 Overfitting 97, 308, 548  
 Oversimplification 305

**P**

Paper prototype 305  
 Parse tree 539  
 Partial program analysis 207  
 PCA. *See* Principal component analysis

PDDP. *See* Principal direction divisive partitioning  
 People 6  
 Per item learning rate 283  
 Per user learning rate 283  
 Perception 286  
 Performance 331  
 Personalization 104, 200, 224, 256, 548  
 Persuasiveness 286, 288, 548  
 Plugin 83, 87, 100, 106, 120, 123, 126, 127  
 Popup 232  
 Positive predictive value. *See* Precision  
 Post-pruning 49  
 Pragmatic reuse 316, 360, 369  
 Precision 117, 127, 141, 154, 161, 189, 250, 252, 276, 282, 310, 472, 548  
 Prediction 132, 250, 302, 548  
 Prediction coverage 253, 253, 283  
 Presenting recommendations 7. *See also* Annotation, Dashboard, Popup, Visualization  
 Principal component analysis (PCA) 51, 62  
 Principal direction divisive partitioning (PDDP) 51  
 Privacy 184, 193, 224, 247, 265, 548  
 Proactive initiation 230  
 Proactive recommendation 102, 110, 230, 319, 339, 548  
 Process metrics 152  
 Producing recommendations 7  
 Product 132  
 Product line 512  
 Product line engineering 511  
 Product metrics 152  
 Productivity 316, 331  
 Profile database 204  
 Program transformation 421, 434, 548  
 Programming-by-demonstration 424  
 Programmer. *See* Developer  
 Programming-by-example 426  
 Progressive disclosure 234  
 Project landscape 2  
 Protocol 121

**Q**

Q-statistic 256  
 Quality 131, 246, 276, 306, 313, 316, 548  
 Query 6, 303–305, 308, 309, 312, 313, 316  
 Quick fix 83, 87

**R**

Radial basis function 60  
 Random forest 49  
 Ratability 258  
 Rating 4  
 Reactive initiation 230  
 Reactive recommendation 110, 230, 339, 548  
 Reactivity 290, 549  
 Recall 117, 127, 152, 154, 161, 189, 250, 276, 282, 287, 310, 483, 549  
 Receiver operator characteristic curve 282  
 Reclassification results 138  
 Recommendation 93, 246, 277, 303, 305, 549  
   box 288, 549  
   environments 31  
   item 2, 15, 253. (*See also* Code example, Command, Issue report, People, Source code) 553  
   task 23  
 Recommendation system 15, 39, 77, 93, 131, 245, 276, 307, 330  
   business aspects 287–290  
   common frameworks for 282  
   in software engineering 2, 5, 39, 93, 199, 223, 302, 330, 478, 549  
   in software engineering, design concerns of. (*See* Capturing context, Data preprocessing, Presenting recommendations, Producing recommendations) 553  
   in-the-small 77, 77–91  
   user aspects 286–287  
 Recommender confidence 257  
 Refactoring 208, 229, 245, 422, 424, 549  
 Reinforcement 81, 81, 86, 549  
 Relevance 78–82, 85–87, 288, 549  
 Reporter 132, 549  
 Repository problem 360  
 Representation problem 360  
 Representative 316  
 Representative developer 325  
 Representativeness 305, 309, 550  
 Reproducibility 324, 550  
 Requirement 15, 238, 265, 291, 550  
 Requirements elicitation/negotiation 455, 550  
 Response time 117, 550  
 Retrieval problem 360  
 Retrospective study 89  
 Reuse cost 317

Reuse-by-memory 373  
 Risk 265  
 Robustness 260, 266, 290, 291, 550  
 Root-mean-squared error 248, 276, 280, 282, 550  
 Runtime 89, 311

**S**

Sample 320  
 Sampling 305  
 Satisfaction 262, 287  
 Satisfiability 513, 550  
 Scalability 118, 247, 266, 281, 285, 290, 291, 550  
 SCoReS. *See* Source code based recommendation system  
 Sensitivity. *See* Recall  
 Serendipity 29, 258, 265, 284, 288, 550  
 Sessionization 186, 186, 193  
 Simple single-pass k-means 70  
 Simulation 116, 301–325, 550  
   environment 304, 304, 305, 308, 309, 312  
   modeling 302  
 Simulator 304, 304  
 Skeleton 315, 315, 316  
 Social network 152, 551  
 Social tagging 113, 324. *See also* Collaborative filtering  
 Software configuration management system. *See* Version control system  
 Software engineer. *See* Developer  
 Software product line 551  
 Software quality metrics 551  
 Software reuse 90, 93, 316, 364  
 Software reuse environment 371  
 Software search 364  
 Software search engine 365  
 Solution for a recommendation task 24  
 Sound. *See* Validity  
 Soundness. *See* Validity  
 Source code 2, 5  
 Source code based recommendation system (SCoReS) 93, 551  
 Spearman's  $\rho$  250  
 Specification 315  
 Specificity 80, 81, 86, 551  
 Speculative analysis 83, 373, 379, 551  
 Stakeholder 306, 455, 551  
 Standalone application 106  
 Stemming 65

Stop word. *See* Stopping  
 Stopping 65  
 Strathcona 314–316  
 Structural context 98  
 Structural fact 315  
 Structural relevance 317  
 Stub 304, 304, 324  
 Suade 80–81  
 Sub-optimal decision 324  
 Supervised learning 41, 486. *See also* C4.5, CART  
 Support 97. *See also* Association rule mining  
 Support count 312, 312, 313  
 Support vector machine (SVM) 60, 488  
 Support-based pruning 43  
 SVM. *See* Support vector machine  
 System constraints 290  
 Systematic change 425  
 Systematic error. *See* Bias

## T

Tangled code changes 148  
 Task 6  
 Task context 6, 175, 203  
 Taxonomy 188, 269, 551  
 Term frequency–inverse document frequency. *See* TF–IDF  
 Test-driven development 551  
 Text link 110, 551  
 Text mining 40, 64–66. *See also* Fayyad–Irani, InfoGain Stemming, Stopping, TF–IDF, Tokenization  
 Text similarity. *See* Textual similarity  
 Textual description 235  
 Textual similarity. *See* Bigram, *n*-gram, Text mining, Unigram  
 TF–IDF 66, 484, 485  
 Theoretical perspective 331  
 Threat to validity. *See* Validity  
 Threshold 81, 84, 312, 313  
 Throughput 263  
 Time series analysis 189  
 Tokenization 64  
 Toolsmith 223, 234, 237, 238  
 Traditional F-measure. *See* F-measure  
 Traditional television 277  
 Training 89, 90, 309, 312, 320  
 Transaction. *See* Commit  
 Transformation rule 422  
 Transformative recommendation 235  
 Transparency 30, 226, 226, 551

Tree learning. *See* Fayyad–Irani, InfoGain, Principal direction divisive partitioning, WHERE  
 Triangulation 303, 307, 308, 321, 324, 552  
 True negative 310. *See* Confusion matrix  
 True negative rate 80, 250, 552  
 True positive 146, 285, 310. *See also* Confusion matrix  
 True positive rate. *See* Recall  
 Trust 88, 228, 246, 256, 265, 286, 324, 341, 552  
 Try it and see 83  
 Type I error. *See* Confusion matrix  
 Type II error. *See* Confusion matrix

## U

Undecidability 5, 311  
 Understandability 104, 225, 225, 552  
 Unigram 486, 487  
 Unit testing 304, 324  
 Unit under test 304  
 Unsupervised learning 40. *See also* Apriori, Expectation maximization, K-means  
 Usability 105, 117, 238, 247, 265, 552  
 Usage history collector 95  
 Usage scenario 313, 323  
 Use case 369  
 Usefulness 284, 294, 314, 318, 320, 321  
 User 84, 95, 103, 223, 246, 456, 552  
   history 108, 552  
   interaction data 3  
   interface 223, 262, 371, 552  
   model 200  
   profile 200  
   satisfaction 284, 553  
 User-based collaborative filtering 16  
 User-space coverage. *See* Prediction coverage  
 Utility 277, 287, 295

## V

Valid. *See* Validity  
 Validation 107  
 Validity 269, 308, 314, 316, 320, 321, 553  
 Variant 515. *See also* Version  
 VCS. *See* Version control system  
 Vector space model 484, 484–486  
 Version 132, 312, 553

Version control system (VCS) 2, 79, 108,  
141, 174, 205, 303, 311, 314, 553  
Visualization 216, 225, 234–236

## W

W2 67

Web 3, 20, 99, 173–175, 177, 179–181, 189,  
190, 192, 194, 277, 287, 288, 290,  
338, 457, 459, 460, 462, 494, 497,  
524

WHERE 51

Wizard of Oz experiment 238, 305, 553

Workspace 212, 231, 239, 303, 304, 308,  
309, 379, 380, 497