**CS256 – Midterm Exam Study Guide**

**By: Zayd Hammoudeh**

**Chapter #04 – Classification: Basic Concepts, Decision Trees, and Model Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification**  Task of assigning objects to one of several predefined categories. | **Training Set**  A collection of records. Each **record** contains a set of attributes one of which is the **class**. | **Model**  A function from the value of record attributes to the class attribute. | **Test Set**  A collection of records used to determine the accuracy of the classification model. | **Example Classification Techniques**   1. **Neural Networks** 2. **Decision Tree** 3. **Rule Based Classifier** 4. **Memory Based Reasoning** 5. **Support Vector Machines** 6. **Naïve Bayes and Bayesian Belief Networks** |

|  |  |  |
| --- | --- | --- |
| **Induction**  Using a training set to generate a model.  **Deduction**  Process of applying a model to a training set.  **Decision Tree Induction**   * **Greedy Strategy** * **Key Decision #1:** Attribute to expand next * **Key Decision #2:** When to stop expanding | **Hunt’s Decision Tree Induction Algorithm:**   * Let *Dt* be the set of training records that reach a node *t*.  1. If *Dt* contains records that **all belong to the same class *yt***, then *t* is a leaf node with class value *yt*. 2. If *Dt* is an **empty set**, then *t* is a leaf node with default value *yd.* 3. If *Dt* contains **records that belong to more than one class and there are no attributes left**, then *t* is a leaf node with default value is a leaf node with default value *yd*. 4. If *Dt* contains **records that belong to more than one class**, then use an attribute test to split the data into smaller subsets. Recursively apply the same procedure above. | **Attribute Types**   * **Binary** – Attribute with exactly two possible values. * **Nominal** – Two or more class values with no intrinsic Order * **Ordinal** – Two or more class values that can be ordered or ranked * **Continuous** – Quantitative attribute that can be measured along a continuum. |

|  |  |  |
| --- | --- | --- |
| **Splitting Nominal and**  **Ordinal Attributes**   * **Binary** – Divides attribute values into two subsets. **This requires the additional step of finding optimal partitioning.** * **Multi-way** – Use as many partitions as distinct values. | **Splitting Based on Continuous Attributes**   * **Discretization** – Form an ordinal categorical attribute.   + **Static** – Discretize once at the beginning   + **Dynamic** – Ranges can be found by equal interval bucketing, equal frequency bucketing, or clustering. * **Binary Decision** (A < v or A >v) – Consider all possible splits and find the best cut.   + **Binary Decision Procedure:** Go between each training set record value and calculate the GINI index if the splitting point was at that value. **Select the splitting point with the lowest GINISPLIT value**.     - **Computationally inefficient** – where is the number of records. | **Homogeneity/Low** **Impurity** – Extent to which nodes in the decision tree have the same class value/distribution.  **Nodes with high levels of homogeneity (i.e. low levels of impurity) are preferred**. |

**Impurity Measures**

**For all of these metrics, a lower score is generally preferable.**

|  |  |  |
| --- | --- | --- |
| **GINI Index**   * – Node in the decision tree * – Class value * – Number of class values * – Probability (i.e. relative frequency) of class value in node   **Minimum Value:** 0 when:  **Maximum Value:** when: | **GINISPLIT**   * – Child node * – Number of records in parent node. Note: * – Number of child nodes (i.e. attribute partitions) * – GINI index value of node .   **Minimum Value:** 0 when:  **Maximum Value:** when: | **Entropy**   * – Node in the decision tree * – Class value * – Number of class values * – Probability (i.e. relative frequency) of class value in node   **Minimum Value:** 0 when:  **Maximum Value:** when: |

|  |  |  |
| --- | --- | --- |
| **Information Gain** | | **Classification Error**   * – Node in the decision tree * – Class value * – Probability (i.e. relative frequency) of class value in node   **Minimum Value:** 0 when:  **Maximum Value:** when: |
| * – Parent node in the decision tree * – Child node in the decision tree * – Number of child nodes * – Number of records in child node * – Number of records in parent node   **Key Note:** A higher is preferable unlike with the other metrics where a lower value was better.  **Disadvantage of Information Gain:** Tends to prefer splits that result in a large number of partitions, each being small but pure (i.e. overfitting) | **Normalizing for Split Size**  **penalizes a large split by reducing the gain.** |

**Stopping Criteria for Decision Tree Induction**

|  |  |  |  |
| --- | --- | --- | --- |
| **Three Stopping Criteria**  **for Decision Tree Induction**   * **When all records in a node have the same class value** * **When all records in a node have similar attribute values**. * **Early Termination** | **Underfitting** – When a model is too simple, both training and test errors are large. | **Overfitting** – When a model becomes too complex (e.g. too large a tree), the test error begins to increase even though the training error decreases.   * **Result:** Training error is **NOT** representative for generalization error. | **Causes of Overfitting**   * **Noise** * **Insufficient training records** (i.e. lack of representative samples) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resubstitution Error**  Error on the **training** set.  **Single Leaf Node Error:**  **Total Resubstitution**  **Error:** | **Generalization Error**  Error on the **testing** data.  **Single Leaf Node Error:**  **Total Generalization**  **Error:** | **Generalization Error Estimation** | | |
| **Optimistic Estimation**  Training error is equal to the testing error. | **Pessimistic Estimation**  Assign a penalty term to ea.  **Total Pessimistic Error**  – Number of leaf nodes. | **Reduced Error Pruning**  Use a validation set to estimate the generalization error. |

|  |  |  |
| --- | --- | --- |
| **Occam’s Razor**  Given two models with similar generalization errors, one should prefer the simpler model over the more complex model.  **This is because more complex model has a greater chance of fitting accidentally by errors in the data.** | **Pre-pruning (Early Stopping Rule)**   * **Stop the induction algorithm before it becomes a full tree.** * **Typical Stopping Rules:**   + All remaining records have the same class value   + All attribute values are the same. * **More restrictive conditions:**   + Number of instances is below a user-specified threshold.   + Expanding the current node does not improve impurity measures (e.g. GINI Index, Information Gain)   + Class distribution of instances are independent of available features. | **Post-pruning (Early Stopping Rule)**   * **Grow the decision tree to its entirety**. * Trim nodes in the tree in a **bottom-up fashion.** * Only **trim nodes if by trimming the estimate of the generalization error improves**. * New leaf node’s **class label is determined from the majority class of instances in the merged node**. |

|  |  |
| --- | --- |
|  |  |

**Handling Missing Attribute Values**

|  |  |  |
| --- | --- | --- |
| **Issues Associated with  Missing Attribute Values**   * **Affects how impurity measures are computed** * **Affects how to distribute instances with missing value to child nodes**. * **Affects how to test instance with missing value is classified**. | **Computing Impurity Measure**   * **Calculate entropies (i.e. information gain) with element with missing value EXCLUDED.** * **Multiply by scalar of elements included over total number of elements** (in below example 9 elements included over 10 total elements hence 0.9): | **Distribute Instances**   * **Split the missing record between the two child nodes** * **Percentage of child node that goes to each child is portion to the relative frequency of that attribute value**. |

|  |  |  |
| --- | --- | --- |
| **Classifying New/Unseen Records with Missing Data**   * **Pick the most likely of child nodes and use continue down that portion of the tree.** | **Data Fragmentation** – At each level of the tree, the number of instances gets smaller. At leaf nodes, the number of instances could be too small to be statistically significant. | **Oblique Decision Tree** – Test condition in a node may involve multiple attributes.   * **Advantage** – Most expressive decision tree * **Disadvantage** – Finding optimal test condition is computationally expensive. |
| **Tree Induction: NP Hard**  **Alternate Strategies**   * **Bottom Up Tree Generation** * **Bidirectional Tree Generation**   + Inside-out Bidirectional   + Outside-in Bidirectional | **Expressiveness** – Decision trees do not generalize well to certain types of functions including a parity function which would require a complete tree. |
| **Decision Boundary** – Borderline between two neighboring regions of different classes. In non-oblique decision trees, this is parallel to access since it involves a single attribute at a time. | **Tree Replication** – In a decision tree, a subtree may appear in multiple branches. This leads to unnecessary memory usage. |

**Performance Evaluation**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * **Focus on the predictive capability of a model.**   **Confusion Matrix**   |  |  |  |  | | --- | --- | --- | --- | |  | **Predicted Class** | | | | **Actual Class** |  | **Class = Yes** | **Class=No** | | **Class = Yes** | **a** | **b** | | **Class=No** | **c** | **d** |   **a** – True Positive (TP)  **b** – False Negative (FN)  **c** – False Positive (FP)  **d** – True Negative (TN) | **Accuracy**   * **Accuracy only tells part of the story**.   + **Example:** Two Class Problem     - Number of Class 0 Examples: 9990     - Number of Class 1 Examples: 10     - If the model predicts everything as class 0, its accuracy is 99.9% but it cannot detect any class 1. | **Cost Matrix**   |  |  |  |  | | --- | --- | --- | --- | |  | **Predicted Class** | | | | **Actual Class** |  | **Class = Yes** | **Class=No** | | **Class = Yes** | C(Y|Y) | C(N|Y) | | **Class=No** | C(Y|N) | C(N|N) |  * – Cost of predicting class “j" given the actual class is “k”. * Cost matrix can be a better performance evaluation as it accounts for different costs of depending on the type of error. |

|  |  |  |
| --- | --- | --- |
| * **Precision** – Accuracy of positive predictions. Biased towards C(Y|Y) & C(Y|N).   + – True positive.   + – False positive. | * **Precision** – Accuracy of records with positive class value. Biased towards C(Y|Y) & C(N|Y).   + – True positive.   + – False negative. | * **F-Measure** – Biased two all except C(N|N) (i.e. true negative)   + – Recall   + – Precision   + – False Positive |

|  |  |  |
| --- | --- | --- |
| – Number of instances covered by rule  – Number of positive instances covered by rule. | **Proportionality of Cost and Accuracy**   * Cost and accuracy are proportional if:   and | **Sample Size and Model Performance**   * **Learning Curve** – Shows how model accuracy changes (and varies) with sample size. * **Effects of Small Sample Size:**   + **Bias in the estimate**   + **Variance in the estimate.** |

**Methods for Model Comparison**

|  |  |  |
| --- | --- | --- |
| **Holdout** – Reserve 2/3 of labelled examples for training and 1/3 for testing.  **Disadvantages**   * **Uses on a subset of the labelled examples** when training the model. * Model **dependent on the composition of the training and test sets**. * **Training and test sets are not independent** since come from same original set. If one class value is over- or under-represented in either set, it will skew the results. | **Random Subsampling** – Repeats the whole out method multiple times with replacement.  **Disadvantages:**   * Still **uses only a subset of the labelled examples** to build the model. * **No control of how many times each record appears in the training and test sets**. If a particular record is always in the training set, it may skew the model.   **Accuracy of *k* Random Subsamplings**   * – Number of iterations * – Accuracy of the iteration. | **Cross Validation** – Partition the labelled dataset into *k* disjoint subsets.   * **k-Fold** – Train on k-1 partitions and test on the remaining one. * **Leave-One-Out** – The number of partitions equals the number of training samples.     **Accuracy of *k*-Fold Cross Validation**   * – Number of iterations * – Accuracy of the iteration.   **Disadvantages:**   * **Computationally expensive** as process is repeated *k* times. * Depending on size of partition (e.g. 1 for Leave-One-Out), **accuracy from iteration to iteration** can vary significantly. |

|  |  |
| --- | --- |
| **Bootstrap** – |  |

|  |  |
| --- | --- |
| **Minimum Description Length** |  |

**Chapter #05 – Additional Classification Techniques**

**Rule-Based Classifiers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifies records using a collection of “if…then…” rules. Form of Rule:**   * **Condition** (**Antecedent**, **LHS**) – Conjunction of attributes. * (**Consequent**, **RHS**) – Class value. | **Cover** – A rule covers an instance if the attributes of satisfy the condition (LHS) of the rule.  **Coverage of a Rule** – Fraction of records that satisfy the antecedent of a rule.  **Accuracy of a Rule** – For records covered by a rule, it is the fraction of records that have the matching class value. | **Mutually Exclusive Rule Set** – Rules in the set are independent of each other such that **each record is covered by at most one rule**.  **Exhaustive Rule Set** – A set of rules that covers every possible combination of attribute values. Hence, **each record is covered by at least one rule**. |  |

|  |  |  |
| --- | --- | --- |
| – Number of instances covered by rule  – Number of positive instances covered by rule. | – Number of instances covered by rule  – Number of positive instances covered by rule.  – Number of classes | – Number of instances covered by rule  – Number of positive instances covered by rule.  – Number of classes  – Prior probability of positive class. |

|  |  |
| --- | --- |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Bootstrap** – | **ROC Curve** | |
| * Used to illustrate the performance of a binary classifier. * Two Dimensional   + **X-Axis** – False Positive Rate   + **Y-Axis** – True Positive Rate |  |
|  | |

**Miscellaneous**

|  |  |  |
| --- | --- | --- |
| **Decision Tree Algorithm**  **Advantages**   * **Inexpensive to construct** * **Extremely fast at classifying unknown records.** * **Easy to interpret for small sized trees.** * **Accuracy is comparable to other classification techniques for many simple datasets.** (Since everything comes right from the data)   **Disadvantages**   * **May not generalize well for certain types of functions** (e.g. Parity function requires a complete tree) * **May be insufficient for modelling continuous variables** that do not allow oblique nodes. |  |  |