Esercitazione 2 – Ereditarietà, Classe astratta, inner class, polimorfismo per inclusione, aggregazione

N.B. Le visibilità di attributi, metodi e classi devono essere scelte appropriatamente dallo studente.

- *Definire una classe astratta **Node** per modellare l'astrazione dell'entità nodo (fogliare o intermedio) dell'albero di decisione*

Membri Attributi

static int *idNodeCount*=0; *// contatore dei nodi generati nell'albero*

int idNode; *// identificativo numerico del nodo*

int beginExampleIndex; *// indice nell'array del training set del primo esempio coperto dal nodo corrente*

int endExampleIndex; *// indice nell'array del training set dell'ultimo //esempio coperto dal nodo corrente. beginExampleIndex e endExampleIndex individuano //un sotto-insieme di training.*

double variance; *// valore della varianza calcolata, rispetto all'attributo di classe, nel sotto-insieme di training del nodo*

Membri Metodi

Node(Data trainingSet, **int** beginExampleIndex, **int** endExampleIndex)

Input: oggetto di classe Data contenente il training set completo, i due indici estremi che identificano il sotto-insieme di training coperto dal nodo corrente

Output://

Comportamento: Avvalora gli attributi primitivi di classe, inclusa la varianza che viene calcolata rispetto all'attributo da predire nel sotto-insieme di training coperto dal nodo

int getIdNode()

Input: //

Output: identificativo numerico del nodo

Comportamento: Restituisce il valore del membro idNode

int getBeginExampleIndex(), **int** getEndExampleIndex()

Input: //

Output: indice del primo (ultimo) esempio del sotto-insieme rispetto al training set complessivo

Comportamento: Restituisce il valore del membro beginExampleIndex (endExampleIndex)

double getVariance()

Input: //

Output: valore della varianza dell'attributo da predire rispetto al nodo corrente

Comportamento: Restituisce il valore del membro variance

abstract int getNumberOfChildren();

Input: //

Output: valore del numero di nodi sottostanti

Comportamento:E' un metodo astratto la cui implementazione riguarda i nodi di tipo test (split node) dai quali si possono generare figli, uno per ogni split prodotto. Restituisce il numero di tali nodi figli.

public String toString()

Input: //

Comportamento: Concatena in un oggetto String i valori di beginExampleIndex,endExampleIndex, variance e restituisce la stringa finale.

- *Definire una classe astratta **SplitNode** per modellare l'astrazione dell'entità nodo di split (continuo o discreto) estendendo la superclasse **Node***

Fornita dal docente

Membri Attributi

class SplitInfo { *// Classe che aggrega tutte le informazioni riguardanti un nodo di split*

Object splitValue; *// valore di tipo Object (di un attributo indipendente) che definisce uno split*

int numberChild; *// numero di split (nodi figli) originanti dal nodo corrente*

String comparator="="; *// operatore matematico che definisce il test nel nodo corrente ("=" per valori discreti)*

SplitInfo(Object splitValue,**int** beginIndex,**int** endIndex,**int** numberChild) *// Costruttore che avvalora gli attributi di classe per split a valori discreti*

SplitInfo(Object splitValue,**int** beginIndex,**int** endIndex,**int** numberChild, String comparator) // *Costruttore che avvalora gli attributi di classe per generici split (da usare per valori continui)*

Object getSplitValue() *// restituisce il valore dello split*

public String toString() // *concatena in un oggetto String i valori di beginExampleIndex,endExampleIndex, child, splitValue, comparator e restituisce la stringa finale.*

String getComparator() *//restituisce il valore dell'operatore matematico che definisce il test*

}

Attribute attribute; *//oggetto Attribute che modella l'attributo indipendente sul quale lo split è generato*

SplitInfo mapSplit[]; *//array per memorizzare gli split candidati in una struttura dati di dimensione pari ai possibili valori di test*

double splitVariance; *//attributo che contiene il valore di varianza a seguuito del partizionamento indotto dallo split corrente*

Membri Metodi

SplitNode(Data trainingSet, **int** beginExampleIndex, **int** endExampleIndex, Attribute attribute)

Input: training set complessivo, indici estremi del sotto-insieme di training, attributo indipendente sul quale si definisce lo split

Output: //

Comportamento: Invoca il costruttore della superclasse, ordina i valori dell'attributo di input per gli esempi beginExampleIndex-endExampleIndex e sfrutta questo ordinamento per determinare i possibili split e popolare l'array mapSplit[], computa la varianza (splitVariance) per l'attributo usato nello split sulla base del partizionamento indotto dallo split

abstract void setSplitInfo(Data trainingSet,**int** beginExampelIndex, **int** endExampleIndex, Attribute attribute);

Input: training set complessivo, indici estremi del sotto-insieme di training, attributo indipendente sul quale si definisce lo split

Output: //

Comportamento:metodo abstract per generare le informazioni necessarie per ciascuno degli split candidati (in mapSplit[])

abstract int testCondition (Object value);

Input: valore dell'attributo che si vuole testare rispetto a tutti gli split

Output: //

Comportamento:metodo abstract per modellare la condizione di test (ad ogni valore di test c'è un ramo dallo split)

Attribute getAttribute()

Comportamento: restituisce l'oggetto per l'attributo usato per lo split

double getVariance ()

Comportamento: restituisce l'information gain per lo split corrente

int getNumberOfChildren()

Comportamento: restituisce il numero dei rami originanti nel nodo corrente;

SplitInfo getSplitInfo(**int** child)

Comportamento: restituisce le informazioni per il ramo in mapSplit[] indicizzato da child.

String formulateQuery()

Comportamento: concatena le informazioni di ciascuno test (attributo, operatore e valore) in una String finale. Necessario per la predizione di nuovi esempi

public String toString()

Comportamento: concatena le informazioni di ciascuno test (attributo, esempi coperti, varianza, varianza di Split) in una String finale.

- *Estendendo la superclasse **SplitNode** definire la classe **DiscreteNode** per modellare l'entità nodo di split relativo ad un attributo indipendente discreto.*

Membri Metodi

public DiscreteNode(Data trainingSet,**int** beginExampelIndex, **int** endExampleIndex, DiscreteAttribute attribute)

Input: training set complessivo, indici estremi del sotto-insieme di training, attributo indipendente sul quale si definisce lo split

Output: //

Comportamento: Istanzia un oggetto invocando il costruttore della superclasse con il parametro attribute

void setSplitInfo(Data trainingSet,**int** beginExampelIndex, **int** endExampleIndex, Attribute attribute)

Input: training set complessivo, indici estremi del sotto-insieme di training, attributo indipendente sul quale si definisce lo split

Output: //

Comportamento (Implementazione da class abstract): istanzia oggetti SpliInfo (definita come inner class in Splitnode) con ciascuno dei valori discreti dell'attributo relativamente al sotto-insieme di training corrente (ossia la porzione di trainingSet compresa tra beginExampelIndex e endExampelIndex*), quindi popola l'array c mapSplit[] con tali oggetti.*

int testCondition (Object value)

Input: valore discreto dell'attributo che si vuole testare rispetto a tutti gli split

Output: numero del ramo di split

Comportamento (Implementazione da class abstract) :effettua il confronto del valore in input rispetto al valore contenuto nell'attributo splitValue di ciascuno degli oggetti SplitInfo collezionati in mapSplit[] e restituisce l'identificativo dello split (indice della posizione nell'array mapSplit) con cui il test è positivo

public String toString()

Comportamento:invoca il metodo della superclasse specializzandolo per discreti

- *Estendendo la superclasse **Node** definire la classe **LeafNode** per modellare l'entità nodo fogliare.*

Membri Attributi

Double predictedClassValue; // valore dell'attributo di classe espresso nella foglia corrente

Membri Metodi

LeafNode(Data trainingSet, **int** beginExampleIndex, **int** endExampleIndex)

Input: training set complessivo, indici estremi del sotto-insieme di training, coperto nella foglia

Output:

Comportamento: istanzia un oggetto invocando il costruttore della superclasse e avvalora l'attributo predictedClassValue (come media dei valori dell'attributo di classe che ricadono nella partizione---ossia la porzione di trainingSet compresa tra beginExampelIndex e endExampelIndex *)*

Double getPredictedClassValue()

Comportamento: restituisce il valore del membro predictedClassValue

int getNumberOfChildren()

Comportamento: restituisce il numero di split originanti dal nodo foglia, ovvero 0.

public String toString()

Comportamento:invoca il metodo della superclasse e assegnando anche il valore di classe della foglia.

- *Definire la classe **RegressionTree** per modellare l'entità l'intero albero di decisione come insieme di sotto-alberi*

Membri Attributi

Node root; // radice del sotto-albero corrente

RegressionTree childTree[] // array di sotto-alberi originanti nel nodo root:vi è un elemento nell'array per ogni figlio del nodo

Membri Metodi

RegressionTree()

Input:

Output: //

Comportamento: istanzia un sotto-albero dell'intero albero

public RegressionTree(Data trainingSet)

Input: training set complessivo

Output: //

Comportamento: istanzia un sotto-albero dell'intero albero e avvia l'induzione dell'albero dagli esempi di training in input **(fornita dal docente)**

boolean isLeaf(Data trainingSet,**int** begin, **int** end,**int** numberOfExamplesPerLeaf)

Input: training set complessivo, indici estremi del sotto-insieme di training, numero minimo che una foglia deve contenere

Output: esito sulle condizioni per i nodi fogliari

Comportamento: verifica se il sotto-insieme corrente può essere coperto da un nodo foglia controllando che il numero di esempi del training set compresi tra begin ed end sia minore uguale di numberOfExamplesPerLeaf.

N.B. isLeaf() è chiamato da learnTree() che è chiamato dal costruttore di RegresioinTree dove numberOfExamplesPerLeaf **è fissato al 10% della dimensione del training set**

SplitNode determineBestSplitNode(Data trainingSet,**int** begin,**int** end)

Input: training set complessivo, indici estremi del sotto-insieme di training,

Output: nodo di split migliore per il sotto-insieme di training

Comportamento: Per ciascun attributo indipendente istanzia il DiscreteNode associato e seleziona il nodo di split con minore varianza tra i DiscreteNode istanziati. Ordina la porzione di trainingSet corrente (tra begin ed end) rispetto all'attributo indipendente del nodo di split selezionato. Restituisce il nodo selezionato.

void learnTree(Data trainingSet,**int** begin, **int** end,**int** numberOfExamplesPerLeaf)

Input: training set complessivo, indici estremi del sotto-insieme di training, numero max che una foglia deve contenere

Output: //

Comportamento: genera un sotto-albero con il sotto-insieme di input istanziando un nodo fogliare (isLeaf()) o un nodo di split. In tal caso determina il miglior nodo rispetto al sotto-insieme di input (determineBestSplitNode()), ed a tale nodo esso associa un sotto-albero avente radice il nodo medesimo (root) e avente un numero di rami pari il numero dei figli determinati dallo split (childTree[]). Ricorsivamente ogni oggetto DecisionTree in childTree[] sarà re-invocato il metodo learnTree() per l'apprendimento su un insieme ridotto del sotto-insieme attuale (begin... end). Nella condizione in cui il nodo di split non origina figli, il nodo diventa fogliare.

Fornita dal docente

String printTree()

Input: //

Output: oggetto String con le informazioni dell'intero albero (compresa una intestazione

Comportamento:

Fornita dal docente

public String toString()

Input: //

Output: oggetto String con le informazioni dell'intero albero

Comportamento: Concatena in una String tutte le informazioni di root-childTree[] correnti invocando i relativi metodo toString(): nel caso il root corrente è di split vengono concatenate anche le informazioni dei rami. Fare uso di per riconoscere se root è SplitNode o LeafNode.

Fornita dal docente

void printRules()

Comportamento: Invoca il metodo di classe toString();

void printRules()

Comportamento: Scandisce ciascun ramo dell'albero completo dalla radice alla foglia concatenando le informazioni dei nodi di split fino al nodo foglia. In particolare per ogni sotto-albero (oggetto DecisionTree) in childTree[] concatena le informazioni del nodo root: se è di split discende ricorsivamente l'albero per ottenere le informazioni del nodo sottostante (necessario per ricostruire le condizioni in AND) di ogni ramo-regola, se è di foglia (leaf) termina l'attraversamento visualizzando la regola.

void printRules(String current)

Input: Informazioni del nodo di split del sotto-albero al livello superiore

Output:

*Comportamento: Supporta il metodo **public void** printRules(). Concatena alle informazioni in current del precedente nodo quelle del nodo root del corrente sotto-albero (oggetto DecisionTree): se il nodo corrente è di split il metodo viene invocato ricorsivamente con current e le informazioni del nodo corrente, se è di fogliare (leaf) visualizza tutte le informazioni concatenate.*

■ *Ridefinire la classe **MainTest** in modo da istanziare in oggetto, eseguire l'apprendimento dell'albero col training set istanziato e stampare le regole e le informazioni dell'albero. Segue un esempio di output:*

Fornita dal docente

```
Output( prova.dat)

********* RULES **********

X=A AND Y=A ==> Class=1.0
X=A AND Y=B ==> Class=1.5
X=B ==> Class=10.0
************************

********* TREE **********
```

```
DISCRETE SPLIT : attribute=X Nodo: [Examples:0-14] variance:255.83333333333331
Split Variance: 0.625
      child 0 split value=A[Examples:0-9]
      child 1 split value=B[Examples:10-14]

DISCRETE SPLIT : attribute=Y Nodo: [Examples:0-9] variance:0.625 Split Variance:
0.0
      child 0 split value=A[Examples:0-4]
      child 1 split value=B[Examples:5-9]

LEAF : class=1.0 Nodo: [Examples:0-4] variance:0.0
LEAF : class=1.5 Nodo: [Examples:5-9] variance:0.0
LEAF : class=10.0 Nodo: [Examples:10-14] variance:0.0

*************************
```

Output (servo.dat)

```
********* RULES **********

pgain=3 AND motor=A ==> Class=4.59999198
pgain=3 AND motor=B ==> Class=4.41998537
pgain=3 AND motor=C ==> Class=3.27998572
pgain=3 AND motor=D ==> Class=1.39999467
pgain=3 AND motor=E ==> Class=2.40001847
pgain=4 AND vgain=1 AND screw=A ==> Class=0.8587544899999999
pgain=4 AND vgain=1 AND screw=B ==> Class=0.701250998
pgain=4 AND vgain=1 AND screw=C ==> Class=0.29062602000000004
pgain=4 AND vgain=1 AND screw=D ==> Class=0.23125061333333333
pgain=4 AND vgain=1 AND screw=E ==> Class=0.33125131333333335
pgain=4 AND vgain=2 AND screw=A ==> Class=0.75375426
pgain=4 AND vgain=2 AND screw=B ==> Class=0.596253146
pgain=4 AND vgain=2 AND screw=C ==> Class=0.3937517375
pgain=4 AND vgain=2 AND screw=D ==> Class=0.38437667499999995
pgain=4 AND vgain=2 AND screw=E ==> Class=0.38437667000000003
pgain=4 AND vgain=3 AND motor=A ==> Class=1.0762503479999999
pgain=4 AND vgain=3 AND motor=B ==> Class=1.031252928
pgain=4 AND vgain=3 AND motor=C ==> Class=0.97125419
pgain=4 AND vgain=3 AND motor=D ==> Class=0.5531278575
pgain=4 AND vgain=3 AND motor=E ==> Class=0.956252188
pgain=5 AND screw=A ==> Class=0.6479201833333333
pgain=5 AND screw=B ==> Class=0.49875247799999994
pgain=5 AND screw=C ==> Class=0.46875225
pgain=5 AND screw=D ==> Class=0.45937718
pgain=5 AND screw=E ==> Class=0.4312519925
pgain=6 AND screw=A ==> Class=0.62812837625
pgain=6 AND screw=B ==> Class=0.4912524260000001
pgain=6 AND screw=C ==> Class=0.46875225
pgain=6 AND screw=D ==> Class=0.45000212
pgain=6 AND screw=E ==> Class=0.4312519925
*************************

********* TREE **********

DISCRETE SPLIT : attribute=pgain Nodo: [Examples:0-166]
variance:403.7886688003771 Split Variance: 164.05369567871094
      child 0 split value=3[Examples:0-49]
      child 1 split value=4[Examples:50-115]
      child 2 split value=5[Examples:116-141]
      child 3 split value=6[Examples:142-166]
```

```
DISCRETE SPLIT : attribute=motor Nodo: [Examples:0-49]
variance:156.80153097461266 Split Variance: 83.47422790527344
      child 0 split value=A[Examples:0-9]
      child 1 split value=B[Examples:10-19]
      child 2 split value=C[Examples:20-29]
      child 3 split value=D[Examples:30-39]
      child 4 split value=E[Examples:40-49]

LEAF : class=4.59999198 Nodo: [Examples:0-9] variance:2.980262085795573
LEAF : class=4.41998537 Nodo: [Examples:10-19] variance:2.896232476843039
LEAF : class=3.27998572 Nodo: [Examples:20-29] variance:18.83591033000802
LEAF : class=1.39999467 Nodo: [Examples:30-39] variance:4.8198987805619815
LEAF : class=2.40001847 Nodo: [Examples:40-49] variance:53.941925798017316
DISCRETE SPLIT : attribute=vgain Nodo: [Examples:50-115]
variance:6.413018878922312 Split Variance: 3.8708453178405 76
      child 0 split value=1[Examples:50-69]
      child 1 split value=2[Examples:70-91]
      child 2 split value=3[Examples:92-115]

DISCRETE SPLIT : attribute=screw Nodo: [Examples:50-69]
variance:1.75247198488759 Split Variance: 0.45010876655578613
      child 0 split value=A[Examples:50-54]
      child 1 split value=B[Examples:55-59]
      child 2 split value=C[Examples:60-63]
      child 3 split value=D[Examples:64-66]
      child 4 split value=E[Examples:67-69]

LEAF : class=0.8587544899999999 Nodo: [Examples:50-54]
variance:0.04949983275139225
LEAF : class=0.701250998 Nodo: [Examples:55-59] variance:0.3245531739857981
LEAF : class=0.29062602000000004 Nodo: [Examples:60-63]
variance:0.006679780500323751
LEAF : class=0.23125061333333333 Nodo: [Examples:64-66]
variance:9.37513000045076E-4
LEAF : class=0.33125131333333335 Nodo: [Examples:67-69]
variance:0.06843845750334909
DISCRETE SPLIT : attribute=screw Nodo: [Examples:70-91]
variance:1.1486026879650941 Split Variance: 0.635493278503418
      child 0 split value=A[Examples:70-74]
      child 1 split value=B[Examples:75-79]
      child 2 split value=C[Examples:80-83]
      child 3 split value=D[Examples:84-87]
      child 4 split value=E[Examples:88-91]

LEAF : class=0.75375426 Nodo: [Examples:70-74] variance:0.30544177951498996
LEAF : class=0.596253146 Nodo: [Examples:75-79] variance:0.1282517871062263
LEAF : class=0.3937517375 Nodo: [Examples:80-83] variance:0.11812663725567385
LEAF : class=0.38437667499999995 Nodo: [Examples:84-87]
variance:0.029180092126402823
LEAF : class=0.38437667000000003 Nodo: [Examples:88-91]
variance:0.05449293825258583
DISCRETE SPLIT : attribute=motor Nodo: [Examples:92-115]
variance:0.9697707080459708 Split Variance: 0.2316684126853943
      child 0 split value=A[Examples:92-96]
      child 1 split value=B[Examples:97-101]
      child 2 split value=C[Examples:102-106]
      child 3 split value=D[Examples:107-110]
      child 4 split value=E[Examples:111-115]

LEAF : class=1.0762503479999999 Nodo: [Examples:92-96]
variance:0.01518537547439891
LEAF : class=1.031252928 Nodo: [Examples:97-101] variance:0.014061019542488395
LEAF : class=0.97125419 Nodo: [Examples:102-106] variance:0.02418647401508256
```

```
LEAF : class=0.5531278575 Nodo: [Examples:107-110] variance:0.07136818444098147
LEAF : class=0.956252188 Nodo: [Examples:111-115] variance:0.10686736068368763
DISCRETE SPLIT : attribute=screw Nodo: [Examples:116-141]
variance:0.4361599353674066 Split Variance: 0.23218290507793427
      child 0 split value=A[Examples:116-124]
      child 1 split value=B[Examples:125-129]
      child 2 split value=C[Examples:130-133]
      child 3 split value=D[Examples:134-137]
      child 4 split value=E[Examples:138-141]

LEAF : class=0.6479201833333333 Nodo: [Examples:116-124]
variance:0.14000193900671576
LEAF : class=0.49875247799999994 Nodo: [Examples:125-129]
variance:0.026437871851308303
LEAF : class=0.46875225 Nodo: [Examples:130-133] variance:0.00843761400038523
LEAF : class=0.45937718 Nodo: [Examples:134-137] variance:0.026367547501228827
LEAF : class=0.4312519925 Nodo: [Examples:138-141] variance:0.030937929001487396
DISCRETE SPLIT : attribute=screw Nodo: [Examples:142-166]
variance:0.4029806128995439 Split Variance: 0.2442690134048462
      child 0 split value=A[Examples:142-149]
      child 1 split value=B[Examples:150-154]
      child 2 split value=C[Examples:155-158]
      child 3 split value=D[Examples:159-162]
      child 4 split value=E[Examples:163-166]

LEAF : class=0.62812837625 Nodo: [Examples:142-149] variance:0.14554888781945996
LEAF : class=0.4912524260000001 Nodo: [Examples:150-154]
variance:0.03543799740174558
LEAF : class=0.46875225 Nodo: [Examples:155-158] variance:0.00843761400038523
LEAF : class=0.45000212 Nodo: [Examples:159-162] variance:0.023906577001118312
LEAF : class=0.4312519925 Nodo: [Examples:163-166] variance:0.030937929001487396

   ************************
```