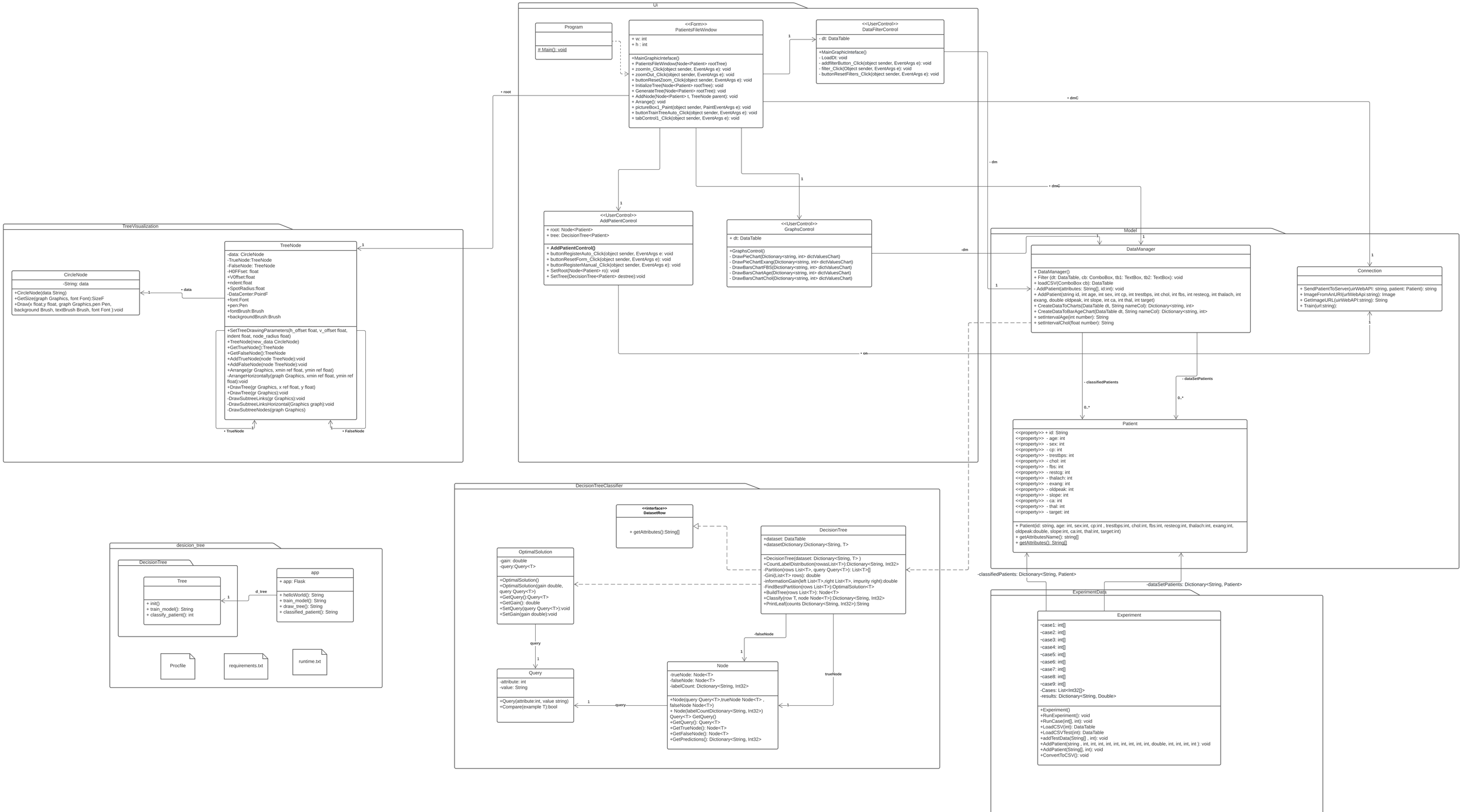
Ui
Program
Main(): void
<<Form>>
PatientsFileWindow
+ w: int
+ h : int
+MainGraphicInteface()
+ PatientsFileWindow(Node<Patient> rootTree)
+ zoomIn_Click(object sender, EventArgs e): void
+ zoomOut_Click(object sender, EventArgs e): void
+ buttonResetZoom_Click(object sender, EventArgs e): void
+ InitializeTree(Node<Patient> rootTree): void
+ GenerateTree(Node<Patient> rootTree): void
+ AddNode(Node<Patient> t, TreeNode parent): void
+ Arrange(): void
+ pictureBox1_Paint(object sender, PaintEventArgs e): void
+ buttonTrainTreeAuto_Click(object sender, EventArgs e): void
+ tabControl1_Click(object sender, EventArgs e): void
<<UserControl>>
DataFilterControl
- dt: DataTable
+MainGraphicInteface()
- LoadDt: void
- addfilterButton_Click(object sender, EventArgs e): void
- filter_Click(Object sender, EventArgs e): void
- buttonResetFilters_Click(object sender, EventArgs e): void
<<UserControl>>
AddPatientControl
+ root: Node<Patient>
+ tree: DecisionTree<Patient>
+ AddPatientControl()
+ buttonRegisterAuto_Click(object sender, EventArgs e: void
+ buttonResetForm_Click(object sender, EventArgs e): void
+ buttonRegisterManual_Click(object sender, EventArgs e): void
+ SetRoot(Node<Patient> ro): void
+ SetTree(DecisionTree<Patient> destree):void
<<UserControl>>
GraphsControl
+ dt: DataTable
+GraphsControl()
- DrawPieChart(Dictionary<string, int> dictValuesChart)
- DrawPieChartExang(Dictionary<string, int> dictValuesChart)
- DrawBarsChartFBS(Dictionary<string, int> dictValuesChart)
- DrawBarsChartAge(Dictionary<string, int> dictValuesChart)
- DrawBarsChartChol(Dictionary<string, int> dictValuesChart)
TreeVisualization
TreeNode
-data: CircleNode
-TrueNode:TreeNode
-FalseNode: TreeNode
-H0FFset: float
+V0ffset:float
+ndent:float
+SpotRadius:float
-DataCenter:PointF
+font:Font
+pen:Pen
+fontBrush:Brush
+backgroundBrush:Brush
+SetTreeDrawingParameters(h_offset float, v_offset float, indent float, node_radius float)
+TreeNode(new_data CircleNode)
+GetTrueNode():TreeNode
+GetFalseNode():TreeNode
+AddTrueNode(node TreeNode):void
+AddFalseNode(node TreeNode):void
+Arrange(gr Graphics, xmin ref float, ymin ref float)
-ArrangeHorizontally(graph Graphics, xmin ref float, ymin ref float):void
+DrawTree(gr Graphics, x ref float, y float)
+DrawTree(gr Graphics):void
-DrawSubtreeLinks(gr Graphics):void
-DrawSubtreeLinksHorizontal(Graphics graph):void
-DrawSubtreeNodes(graph Graphics)
CircleNode
-String: data
+CircleNode(data String)
+GetSize(graph Graphics, font Font):SizeF
+Draw(x float,y float, graph Graphics,pen Pen, background Brush, textBrush Brush, font Font):void
• data
• TrueNode
• FalseNode
• root
• dmC
- dm
-dm
• dmC
• cm
Model
DataManager
+ DataManager()
+ Filter (dt: DataTable, cb: ComboBox, tb1: TextBox, tb2: TextBox): void
+ loadCSV(ComboBox cb): DataTable
+ AddPatient(attributes: String[], id:int): void
+ AddPatient(string id, int age, int sex, int cp, int trestbps, int chol, int fbs, int restecg, int thalach, int exang, double oldpeak, int slope, int ca, int thal, int target)
+ CreateDataToCharts(DataTable dt, String nameCol): Dictionary<string, int>
+ CreateDataToBarAgeChart(DataTable dt, String nameCol): Dictionary<string, int>
+ setIntervalAge(int number): String
+ setIntervalChol(float number): String
Connection
+ SendPatientToServer(uirWebAPI: string, patient: Patient): string
+ ImageFromAnURI(urlWebApi:string): Image
+ GetImageURL(uirWebAPI:string): String
+ Train(url:string):
- classifiedPatients
- dataSetPatients
0..*
0..*
Patient
<<property>> + id: String
<<property>> - age: int
<<property>> - sex: int
<<property>> - cp: int
<<property>> - trestbps: int
<<property>> - chol: int
<<property>> - fbs: int
<<property>> - restcg: int
<<property>> - thalach: int
<<property>> - exang: int
<<property>> - oldpeak: int
<<property>> - slope: int
<<property>> - ca: int
<<property>> - thal: int
<<property>> - target: int
+ Patient(id: string, age: int, sex:int, cp:int , trestbps:int, chol:int, fbs:int, restecg:int, thalach:int, exang:int, oldpeak:double, slope:int, ca:int, thal:int, target:int)
+ getAttributesName(): string[]
+ getAttributes(): String[]
-classifiedPatients: Dictionary<String, Patient>
-dataSetPatients: Dictionary<String, Patient>
ExperimentData
Experiment
-case1: int[]
-case2: int[]
-case3: int[]
-case4: int[]
-case5: int[]
-case6: int[]
-case7: int[]
-case8: int[]
-case9: int[]
-Cases: List<Int32[]>
-results: Dictionary<String, Double>
+Experiment()
+RunExperiment(): void
+RunCase(int[], int): void
+LoadCSV(int): DataTable
+LoadCSVTest(int): DataTable
+addTestData(String[] , int): void
+AddPatient(string , int, int, int, int, int, int, int, int, int, double, int, int, int, int, int): void
+AddPatient(String[], int): void
+ConvertToCSV(): void
desicion_tree
DecisionTree
Tree
+ init()
+ train_model(): String
+ classify_patient(): int
d_tree
app
+ app: Flask
+ helloWorld(): String
+ train_model(): String
+ draw_tree(): String
+ classified_patient(): String
Procfile
requirements.txt
runtime.txt
DecisionTreeClassifier
<<interface>>
DatasetRow
+ getAttributes():String[]
DecisionTree
+dataset: DataTable
+datasetDictionary:Dictionary<String, T>
+DecisionTree(dataset: Dictionary<String, T>)
+CountLabelDistribution(rowasList<T>):Dictionary<String, Int32>
-Partition(rows List<T>, query Query<T>): List<T>[]
-Gini(List<T> rows): double
-informationGain(left List<T>,right List<T>, impurity right):double
-FindBestPartition(rows List<T>):OptimalSolution<T>
+BuildTree(rows List<T>): Node<T>
+Classify(row T, node Node<T>):Dictionary<String, Int32>
+PrintLeaf(counts Dictionary<String, Int32>):String
OptimalSolution
-gain: double
-query:Query<T>
+OptimalSolution()
+OptimalSolution(gain double, query Query<T>)
+GetQuery():Query<T>
+GetGain(): double
+SetQuery(query Query<T>):void
+SetGain(gain double):void
query
Query
-attribute: int
-value: String
+Query(attribute:int, value string)
+Compare(example T):bool
query
Node
-trueNode: Node<T>
-falseNode: Node<T>
-labelCount: Dictionary<String, Int32>
+Node(query Query<T>,trueNode Node<T> , falseNode Node<T>)
+ Node(labelCountDictionary<String, Int32>) Query<T> GetQuery()
+GetQuery(): Query<T>
+GetTrueNode(): Node<T>
+GetFalseNode(): Node<T>
+GetPredictions(): Dictionary<String, Int32>
-falseNode
trueNode
1