НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ КОРПОРАЦИЯ ИТМО

Факультет программной инженерии и компьютерной техники

Системы искусственного интеллекта

Лабораторная работа № 3

Вариант №2 (для нечетного порядкового номера)

Выполнил
студент

Нестеров Иван Алексеевич

Группа P33302

Преподаватель: Королёва Юлия Александровна

г. Санкт-Петербург

2022

Цель задания:

1. Дан датасет с данными про оценки студентов инженерного и педагогического факультетов. Для данного датасета необходимо ввести метрику: студент успешный или же нет, которую определять на основании грейда;
2. Отобрать случайным образом sqrt(n) признаков;
3. Реализовать без использования сторонних библиотек построение дерева решений;
4. Провести оценку реализованного алгоритма с использованием accuracy, precision и recall;
5. Построить AUC-ROC и AUC-PR.

Ход работы:

В ходе работы был написан создан maven-проект и средствами языка Java был написан весь необходимый код. Из сторонних библиотек (что можно увидеть в содержании тега dependencies pom.xml файла) использовался только Lombok для генерации конструкторов, геттеров, сеттеров, а также JavaFX FXML и JFRE Chart для построения графиков из 5-го пункта задания. Код и дубликат данного отчета опубликованы на GitHub. Для понятности ко всем классам, конструкторам, методам оформлена документация.

Ссылка:

Построенное дерево принятия решений:

COURSE ID

|---1:

| 29

| |---1:

| | Result: 0

| |---2:

| | Result: 0

| |---3:

| | 23

| | |---1:

| | | 25

| | | |---2:

| | | | 30

| | | | |---2:

| | | | | Result: 1

| | | | |---3:

| | | | | Result: 0

| | | |---3:

| | | | Result: 0

| | |---2:

| | | Result: 0

| | |---3:

| | | Result: 1

| |---4:

| | 5

| | |---1:

| | | Result: 0

| | |---2:

| | | 10

| | | |---1:

| | | | 30

| | | | |---1:

| | | | | Result: 0

| | | | |---2:

| | | | | 2

| | | | | |---1:

| | | | | | Result: 0

| | | | | |---2:

| | | | | | Result: 1

| | | | |---3:

| | | | | Result: 0

| | | | |---4:

| | | | | Result: 0

| | | |---2:

| | | | Result: 1

| | | |---3:

| | | | Result: 1

| |---5:

| | 23

| | |---1:

| | | 10

| | | |---1:

| | | | 14

| | | | |---1:

| | | | | 5

| | | | | |---1:

| | | | | | Result: 0

| | | | | |---2:

| | | | | | Result: 1

| | | | |---2:

| | | | | Result: 0

| | | |---2:

| | | | Result: 0

| | |---2:

| | | Result: 1

| | |---3:

| | | Result: 0

|---2:

| 3

| |---2:

| | Result: 1

| |---3:

| | Result: 0

|---3:

| Result: 1

|---4:

| 20

| |---1:

| | Result: 0

| |---2:

| | Result: 1

|---5:

| 23

| |---1:

| | 2

| | |---1:

| | | Result: 0

| | |---2:

| | | Result: 0

| |---3:

| | Result: 1

|---6:

| 5

| |---1:

| | 25

| | |---2:

| | | Result: 1

| | |---3:

| | | Result: 0

| |---2:

| | Result: 1

|---7:

| 2

| |---1:

| | 23

| | |---1:

| | | Result: 0

| | |---2:

| | | Result: 1

| |---2:

| | Result: 1

|---8:

| Result: 0

|---9:

| 14

| |---1:

| | Result: 0

| |---2:

| | Result: 1

Полученные метрики:

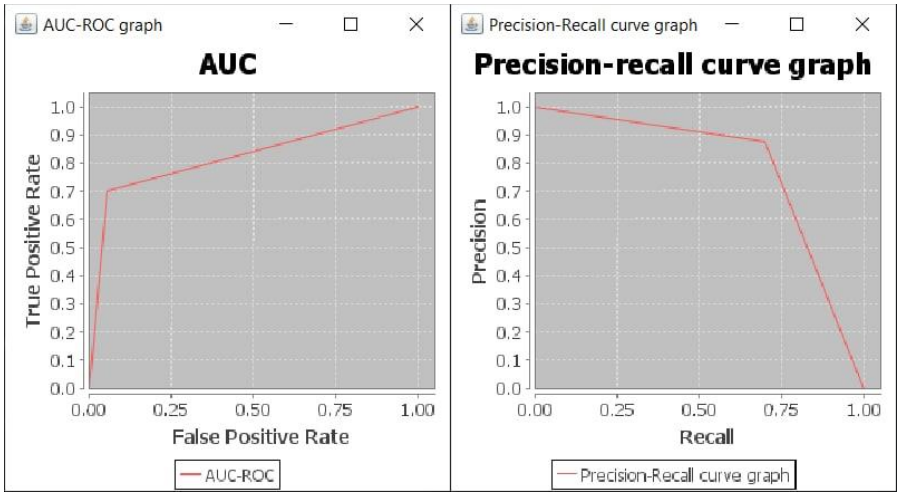
Key class: 1

Precision: 0.875

Recall: 0.700

Accuracy: 0.857

Построенные графики:



Пояснение к алгоритму построения дерева решений:

За основу был взят алгоритм C4.5. Для начала выбирается атрибут, по которому будет построен узел на основе постепенного прироста информации. Итак, на первом шаге имеется корень и некоторое множество T, которое необходимо разбить на несколько подмножеств. Выбранный ранее атрибут будет использоваться в качестве проверки. Пусть этот атрибут будет называться A и иметь n значений, что даёт разбиение на n подмножеств. Далее, что логично, создаются n потомков корня, каждому из которых поставлено в соответствие некоторое своё подмножество, полученное при указанном разбиении. Процедура выбора атрибута и разбиения по нему рекурсивно применяется к «созданным» потомкам и останавливается в двух случаях:

1. После ветвления в вершине было образовано подмножество, все элементы которого принадлежат одному классу. В таком случае вершина становится листом, а этот самый класс – решением листа.
2. Разбивать на подмножества стало нечего. В таком случае вершина оказалась ассоциированной с пустым множеством. Следовательно, она становится листом, а в качестве решения будет выбран наиболее часто встречающийся класс у родителя данного листа.

Рекурсивное завершение разбиений вершин дерева по двум вышеуказанным пунктам позволит сделать вывод о завершении построения дерева целиком.

Исходный датасет будет поделен программой в отношении 4:1 на тренировочную и тестовую выборку. Для построения дерева по указанному принципу будет выбрано случайным образом sqrt(n) атрибутов (n – количество строк в датасете).

Вывод: по ходу выполнения лабораторной работы был реализован алгоритм построения дерева решений на основе алгоритма C4.5. Представленная программа отображает структуру построенного дерева решений, рассчитывает метрики accuracy, precision, recall при тестировании полученного дерева. Кроме того, при помощи библиотеки JFRE Chart реализована возможность построения графиков AUC-ROC и AUC-PR.