Data Mining

AmirHossein Vatanibaf

December 30, 2020

1 Introduction

”Statistical thinking will one day be as necessary for efficient citizenship as is the ability to read and write”
H.G. Wells Circa 1925

2 Data Mining

Data mining is the process of finding anomalies, patterns and correlations within large data sets to predict outcomes. Using a broad range of techniques, you can use this information to increase revenues, cut costs, improve customer relationships, reduce risks and more.

Table 1: Data Mining Techniques

	Predictive	**Discriptive**
1.	Classification	Clustering
2.	Regression	Summerization
3.	Time Series Analysis	Association Rule Mining
4.	Predict	Feature Extraction

3 formulas

Linear Regression

$$f(x) = \sum_{i=1}^{n} m_i x_i + b$$

K Nearest Neghibour

$$D(x,y) = \sqrt{(\sum_{i=1}^{n} (x_i - y_i)^2)}$$

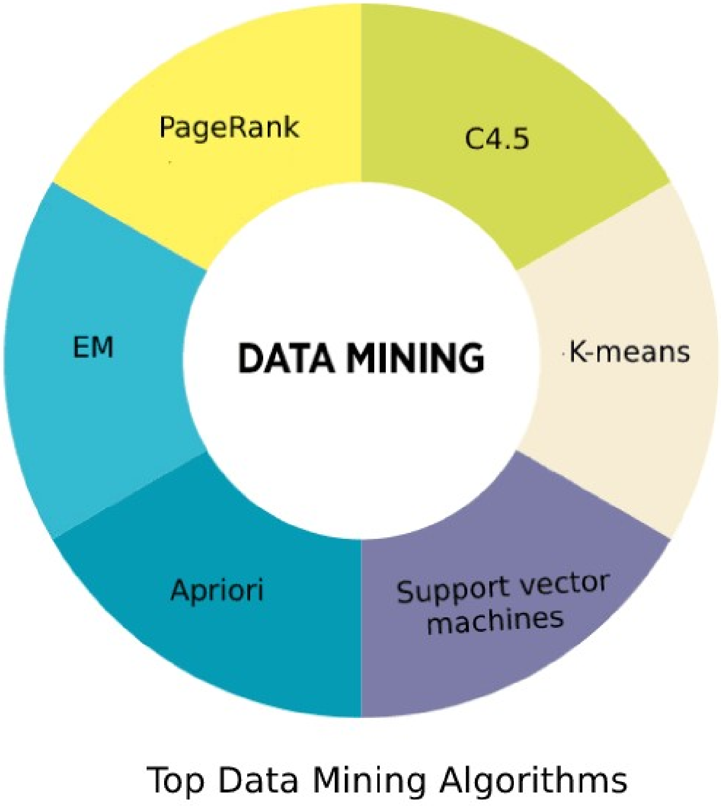

Figure 1: A List Of Top Data Mining Algorithms

4 Code Example with cpp,Program BFS Algorithm

```
#include<iostream>
#include<vector>
#include<algorithm>

using namespace std;

struct Edge
{
        char Vertex1;
        char Vertex2;
};

void InPut(vector<Edge> &Graph)
{
        int Size;
        cout << "Size of Edges :";
        cin >> Size;
        for (int i = 0; i < Size; i++)
        {
                Edge temp;
                cout << "Edge[" << i + 1 << "] :\n";
                cout << "frist Vertex :";
                cin >> temp.Vertex1;
                cout << "secound Vertex :";
                cin >> temp.Vertex2;
                Graph.push_back(temp);
        }
}

void Print(vector<char> Graph)
{
        cout << "\n";
        for (int i = 0; i < Graph.size(); i++)
                cout << Graph[i];

        cout << endl;
}

bool Search(vector<char> Graph, char Key)
{
        for (int i = 0; i < Graph.size(); i++)
                if (Graph[i] == Key)
```

```
                        return true;

        return false;
}

void BFS(vector<Edge> Graph)
{
        char Start;
        vector<char> Queue;
        cout << ”\nStart Vertex :”;
        cin >> Start;
        Queue.push_back(Start);
        for (int i = 0; i < Queue.size(); i++)
        {
                vector<char> Neighbors;
                for (int j = 0; j < Graph.size(); j++)
                {
                        if (Queue[i] == Graph[j].Vertex1)
                                Neighbors.push_back(Graph[j].Vertex2);

                        if (Queue[i] == Graph[j].Vertex2)
                                Neighbors.push_back(Graph[j].Vertex1);

                }
                sort(Neighbors.begin(), Neighbors.end());
                for (int j = 0; j < Neighbors.size(); j++)
                        if (!Search(Queue, Neighbors[j]))
                        Queue.push_back(Neighbors[j]);
        }
        Print(Queue);
}

int main()
{
        vector<Edge> Graph;
        InPut(Graph);
        BFS(Graph);
}
```