به نام خدا

محمد جواد امین – ۹۵۲۳۰۰۸ **گزارش کار سری اول AP**

در این گزار ش کار در هر مرحله بر اساس صورت مسئله پیش می رویم.

در ابتدا یک فایل اکسل داریم با مجموعه ایی از داده ها که باید آن را بخوانیم.

```
std::vector<std::vector<double>> getData(const char* ch)
{
    std::ifstream in_file{ ch };
    double data{};
    std::string temp{};
    std::vector<std::vector<double>> data_vector;

    for (size_t i = 0; i < 234; i++)
    {
        std::vector<double> row{ 1 };

        for (size_t j = 0; j < 6; j++)
        {
            std::getline(in_file, temp, ',');
            data = std::stod(temp);
            if (j == 2 || j == 3)
                data = data / 100.;
            row.push_back(data);
        }
        in_file >> temp;
        data = std::stod(temp);
        row.push_back(data);
        data_vector.push_back(row);
    }
    return data_vector;
}
```

طبق صورت مسئله prototype تابع را در ابتدا تعریف میکنیم . سپس تابع را به صورت شکل بالا تعریف میکنیم.

ورودی تابع const char* است که یک رشته را ورودی میگیرد که اسم فایل اکسل است.

سپس یک متغییر ifstream برای کار با فایل یک متغییر و string ,double برای گرفتن اطلاعات از ورودی است .سپس یک وکتور ۲ بعدی برای گرفتن data و ذخیره اطلاعات تعریف میکنیم.

حال دو for داریم که for اول برای سطر ها و for دوم برای ستون ها است. در هرگام for اول یک وکتور که اولین داده آن همیشه ۱ است تعریف میکنیم. و سپس اطلاعات هر دانشجو رو در وکتور مذکور

ذخیره میکنیم و در آخر ؛ کل اطلاعات هر دانشجو را در وکتور دو بعدی که در ابتدا تعریف کردیم ذخیره میکنیم.

نکته قابل توجه این است که ما برای گرفتن اطلاعات از دستور getline استفاده میکنیم. که اعداد را به صورت رشته میگیرد و برای تبدیل آن به double از دستور std::stod استفاده می شود و در آخر رشته مرتبط با هر دانشجو را در وکتور دو بعدی اصلی ذخیره میکنیم.

نکته مهم دیگر داده‌هایی که مقدار آن ها بیش تر از یک است یعنی ساعاتی که هر دانشجو در هفته مطالعه وکد میزند را ما نرمالایز میکنیم که در مراحل بعد به مشک نخوریم چون در غیر این صورت تعداد iteration ها را زیادکرده و همچنین مقدار learning rate را نیز باید خیلی کم کرد پس برای پرهیز از این اقدامات که باعث کند شدن برنامه ما می شود ؛ داده ها را نرمالایز میکنیم که ما در این قسمت تقسیم بر ۱۰۰ میکنیم.

و در آخر هم تابع وکتور شامل همه ویژگی های تمام دانشجو ها را که تمام متغییر های آن double است باید برگرداند.

در قدم دوم باید داده هایی که در تابع getdata در قسمت قبل از فایل اکسل گرفتیم را به صورت مناسب و همانطورکه در صورت سوال گفته شده ؛ نمایش داده شود. کد این قسمت به صورت زیر است.

```
void displayDataset(std::vector <std::vector <double >> v)
{
    std::cout << "Bias" << std::setw(17) << "Class" << std::setw(19) << "TA" << std::setw(23) << "Coding" << std::setw(21) << "Studying" << std::setw(20) << "Back ground";
    std::cout << std::setw(17) << "Mind" << std::setw(20) << "Grade" << std::endl;
    for (size_t i = 0; i < 36 ; i++)
        std::cout << "****";
    std::cout << std::endl;

    for (size_t i = 0; i < 234; i++)
    {
        for (size_t j = 0; j < 8; j++)
        {
            if (j == 3 || j == 4)
                v[i][j] = 100 * v[i][j];
            if (j == 7)
                std::cout << v[i][j] << std::endl;
            else
                std::cout <<v[i][j]<< std::setw(20);
        }
    }
}
```

که در این قسمت برای تعیین فاصله ها و ... از دستور std::setw() استفاده شده همان طوری که در کد بالا نمایش داده شده است و برای نمایش داده ها هم از دو for تو در تو استفاده میکنیم.

باید توجه داشت که داده هایی که ما در قسمت قبل نرمالایز کرده ایم در این جا باید در ۱۰۰ ضرب کنیم.

شکل خروجی به صورت زیر است.

Bias	Class	TA	Coding	Studying	Back ground	Mind	Grade
1	0.42	0.83	13	28	0.76	0.48	14.23
1	0.47	0.54	68	40	0.39	0.4	15.76
1	0.7	0.06	48	1	0.28	0.29	9.99
1	1	0.07	35	4	0.73	0.49	13.39
1	0.03	0.76	69	22	0.87	0.12	11.26
1	0.72	0.87	26	7	0.25	0.06	12.74
1	0.13	0.77	26	20	0.32	0.46	10.11
1	0.72	0.92	21	7	0.48	0.94	14.7
1	0.34	0.43	46	23	0.41	0.95	13.34
1	0.38	0.4	58	25	0.28	0.59	12.92
1	0.93	0.47	35	33	0.72	0.01	17.72
1	0.91	0	60	10	0.25	0.62	12.81
1	0.67	0.67	52	32	0.27	0.09	16.03
1	0.64	0.9	13	18	0.85	0.64	14.98
1	0.03	0.12	6	39	0.95	0.65	9.51
1	0.71	0.8	12	35	0.38	0.85	17.95
1	0.04	0.63	34	3	0.61	0.23	7.83
1	0.96	0.79	59	37	0.11	0.55	20
1	0.09	0.25	23	32	0.72	0.49	8.91
1	0.09	0.48	33	37	0.14	0.44	10.27
1	0.98	0.59	31	36	0.34	0.26	18.62
1	0.68	0.22	56	13	0.47	0.45	13.31
1	0.2	0.09	54	20	0.8	0.04	8.1
1	0.52	0.18	37	30	0.89	0.06	12.34
1	0.88	0.82	55	24	0.81	0.22	19.48
1	0.2	0.05	50	22	0.9	0.59	9.27
1	0.32	0.56	36	3	0.19	0.31	8.49
1	0.52	0.72	43	18	0.73	0.19	13.74
1	0.65	0.78	67	33	0.42	0.01	17.41
1	0.14	0.31	64	7	0.11	0.99	8.46
1	0.82	0.79	27	38	0.04	0.93	19.96
1	0.87	0.61	41	12	0.61	0.03	15.97
1	0.07	0.92	25	2	0.51	0.64	9.03

توجه شود که ورودی این تابع یعنی display data همان وکتور دو بعدی است که از تابع getdata از قسمت قبل بدست آمده ؛ می باشد.

قسمت سوم ؛ تابع estimator function است که دو ورودی دارد که هر دو به صورت وکتور بوده که یکی وزن ها و یکی ویژگی هر دانشجو است. که هر وزن باید در ویژگی ضرب شود و نمره را بدهد. که کد این تابع به صورت زیر است.

$$grade(\mathbf{x}) = x_0 w_0 + x_1 w_1 + \ldots + x_6 w_6$$

```
double grade(std::vector<double> w, std::vector<double> x)
{
    double grade_student{};

    for (size_t i = 0; i < 7; i++)
        grade_student += x[i] * w[i];

    return grade_student;
}
```

در مرحله بعد cost function را طبق این فرمول زیرکه در صورت مسئله است ؛ تعریف میکنیم.

$$J(\mathbf{w}) = \frac{1}{2m}\sum_{i=1}^{m}(grade(\mathbf{x}^{(i)}) - y^{(i)})^2$$

سپس یک for به تعدادگام تعداد دانشجو ها تعریف کرده که در هرگام اطلاعات هرکدام از آن ها در یک وکتور ذخیره می شود و cost function را طبق فرمول بالا محاسبه میکند . کد این بخش در پایین قابل مشاهده است.

```
double J(std::vector<double> w, std::vector<std::vector<double>> data )
{
    double cost{};

    for (size_t i = 0; i < 234; i++)
    {
        std::vector<double> x{};

        for (size_t j = 0; j < 7; j++)
            x.push_back(data[i][j]);

        cost += std::pow(grade(w, x) - data[i][7], 2);
    }

    return cost/(2*234.);
}
```

قابل توجه اینکه که همه داده های ما به خاطر نرمالایزی که در قسمت های قبل کردیم بین ۰ تا ۱ است.

در این قسمت به سراغ پیدا کردن وزن های هر ویژگی دانشجو می‌رویم که بتوانیم با استفاده از آن وزن ها که بدست می آوریم نمره هر دانشجوی جدید را با داشتن ویژگی‌هایش پیش بینی کنیم.

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial grade}\frac{\partial grade}{\partial w_i} = \frac{1}{m}\sum_{j=1}^{m}(grade(\mathbf{x}^{(j)}) - y^{(j)})x_i^{(j)}$$

$$w_i := w_i - \alpha\frac{\partial J}{\partial w_i}$$

فرمول بالا مشتق تابع cost function است که قبلا تابع آن را نوشتیم. و حالا باید در هر iteration ؛مشتق حساب شده طبق فرمول بالا محاسبه شود و سپس وزن ها با توجه به مقدار مشتق ؛به روز رسانی می گردد و مجددا در iteration بعدی این روند تکرار شود.

حالا به کد پیاده سازی شده این مرحله و شرح آن در زیر می‌پردازیم.

```
std::vector<double> train(std::vector<std::vector<double>> data, std::vector<double> w, double alpha, size_t max_iters, double min_cost, bool verbose)
{
    std::vector<double> w_i{ w };

    for (size_t step{}; step < max_iters; step++)
    {
        std::vector<double> derivative(7,0);

        for (size_t i = 0; i <234 ; i++)
        {
            std::vector<double> x{};
            for (size_t j = 0; j < 7; j++)
            {
                x.push_back(data[i][j]);
            }
            for (size_t j = 0; j < 7; j++)
                derivative[j] += ((grade(w_i, x) - data[i][7]) * x[j]);
        }
        if (verbose == 1)
            std::cout << "iteration " << std::setw(6) << step + 1 << " : the last value of the cost function is : " << std::setw(10) << J(w_i, data) << std::endl;
        for (size_t i{}; i < 7; i++)
            w_i[i] = w_i[i] - (alpha * (derivative[i])/234.);
        if (verbose == 1)
            std::cout << "iteration " << std::setw(6) << step + 1 << " : the current value of the cost function is : " << std::setw(7) << J(w_i, data) << std::endl;
        if(verbose==1)
            std::cout << std::endl;
        if (J(w_i, data) < min_cost)
            break;
    }

    return w_i;
```

همانطورکه در صورت مسئله مشخص شده ؛ ورودی های تابع شامل :یک وکتور دو بعدی (که همان data است) و یک وکتورکه شامل وزن های اولیه و یک double که مقدار learning rate آن و مقادیر بعدی Maximum iterations و مقدار min cost و مقدار verbose است.

سپس در پیاده سازی ابتدا یک وکتور تعریف کرده که مقدار اولیه آن مقدار اولیه وزن‌هایی است که به ورودی تابع داده‌ایم و یک for به تعدادگام های iteration ها تعریف کرده. و در هر iteration فرمول بالا را حساب می‌کنیم که این کار را با یک for انجام میدهم فقط باید توجه کرد برای هر دانشجو هر دفعه با یک وکتورکه داخل for است ویژگی های آن ها را میگریم و داخل وکتور میریزیم چون باید آن را به ورودی تابع grade بدهیم و این کار را برای هر دانشجو تکرار میکنیم.

بعد محاسبه فرمول اول به کمک فرمول دوم که در بالا ذکر شده وزن ها را برای iteration بعدی آپدیت میکنیم.

حال با توجه به مقدار verbose از ورودی تابع اگر مقدار آن true بود باید در هر iteration مقدار تابع cost function را بعد و قبل از آپدیت کردن وزن ها چاپ کندکه این کار با دو if به سادگی به صورت بالا قابل انجام است.

و نکته آخر این که اگر در هر iteration مقدار تابع cost function کمتر از min cost که به ورودی تابع داده ایم شد؛ iteration ها متوقف شود و وزن را در همان iteration برگرداند. که با یک if ساده قابل انجام است.

در این قسمت ؛بعد از بدست آوردن وزن ها از تابع train ؛ نمره‌های هر ۲۳۴ نفر دانشجو را با استفاده از تابع grade محاسبه و چاپ می‌نماییم و در مقابل آن مقدار واقعی نیز چاپ می‌گردد.

```
void displaying_output(std::vector<double> w, std::vector<std::vector<double>> data)
{
    std::cout << "  No" << std::setw(25) << "Real Grade" << std::setw(23) << "Estimated Grade" << std::endl;
    for (size_t i = 0; i < 28; i++)
        std::cout << "**";
    std::cout << std::endl;

    for (size_t i = 1; i <= 234; i++)
    {
        std::vector<double> x{};
        for (size_t j = 0; j < 7; j++)
        {
            x.push_back(data[i-1][j]);
        }
        std::cout <<std::setw(4)<< i << std::setw(20)<< data[i-1][7] << std::setw(20) << grade(w, x)<<std::endl;
    }
}
```

مقادیر بالا را با توجه به فرمتی که در صورت مسئله نمایش داده شده چاپ می‌گردد وکه این کار با یک for که به تعداد ۲۳۴ گام (تعداد دانشجو ها) است انجام می‌گردد و برای هر دانشجو نیز باید در داخل یک for ویژگی های آن دانشجو را در داخل یک وکتور ذخیره شود تا به ورودی تابع grade بدهیم و در آخر این مقدار تابع grade و نمره واقعی را به صورت مناسب چاپ میکنیم .همانگونه که روشن است ورودی این تابع که اسم آن displaying _output است یک وکتور است که شامل وزن های تابع train است و ورودی دوم یک وکتور دو بعدی که data است. خرجی تابع به صورت زیر است.

```
  No                Real Grade         Estimated Grade
**************************************************************
   1                    14.23                 14.045
   2                    15.76                14.0102
   3                     9.99                11.2019
   4                    13.39                13.8976
   5                    11.26                12.9013
   6                    12.74                 13.194
   7                    10.11                11.5385
   8                     14.7                15.8906
   9                    13.34                13.2896
  10                    12.92                12.5899
  11                    17.72                14.6135
  12                    12.81                13.2334
  13                    16.03                13.7185
  14                    14.98                15.6782
  15                     9.51                10.5307
  16                    17.95                15.3595
  17                     7.83                10.5856
  18                       20                16.5344
  19                     8.91                10.6758
  20                    10.27                10.4057
  21                    18.62                14.9072
  22                    13.31                13.0848
  23                      8.1                10.3334
  24                    12.34                12.1605
  25                    19.48                16.7773
  26                     9.27                11.6687
  27                     8.49                10.8334
  28                    13.74                13.9722
  29                    17.41                14.6822
  30                     8.46                11.4129
```

در قسمت بعد ؛ باید یک تابع بنویسم که با فراخوانی آن وزن ها ؛ یک اسم از ورودی بگیرد و وزن های داده شده را در یک فایل اکسل ذخیره کند.

```
void save(std::vector<double> w, const char* name)
{
    std::ofstream out_file{ name };

    for (size_t i = 0; i < w.size(); i++)
        out_file << w[i]<<",";
    out_file.close();
}
```

که این کار با استفاده از تعریف یک متغیر std::ofstream و یک for به راحتی که درکد بالا قابل مشاهده است؛ قابل انجام است.

سپس ؛ یک تابع می‌نویسم که با فراخوانی آن وزن هایی که در داخل فایل اکسل ذخیره شده را بخواند و در داخل یک وکتور ذخیره کرده و آن را برگرداند.

```
std::vector<double> load(const char* name)
{
    std::ifstream in_file{ name };
    std::vector<double> w{};
    double temp{};
    char ch;

    for (size_t i = 0; i < 7; i++)
    {
        in_file >> temp>>ch ;
        w.push_back(temp);
    }
    in_file.close();

    return w;
}
```

ورودی های این تابع اسم فایل اکسل است.

پیاده سازی این تابع نیز به صورت رو به رو است که با یک متغیر std::ifstream و با استفاده از یک for که در هر مرحله یک متغیر double و یکchar میگرد .

متغیر char برای این است که اعداد در این نوع فایل های با ’,‘ از هم جدا می شوند و برای گرفتن کاما ها ما از متغیر ch استفاده میکنیم.

در آخرین مرحله باید تابع predict را بنویسیم که ورودی آن وکتور وزن ها است که از تابع train بدست آمده ودر داخل خود تابع ؛ کاربر ویژگی ها را وارد می‌کند و داخل یک وکتور ذخیره می‌شود سپس باید دوکتور که یکی وزن ها و یکی ویژگی ها هستند را به تابع grade به عنوان ورودی می‌دهیم که آن هم نمره ی تخمینی را برگرداند.

نکته مهم این است که اعداد وارده باید در محدوده مشخصی باشند و اگر خارج از آن محدوده باشد ممکن است نمره بیم صفر تا بیست تولید نشود پس با دو تابع try و catch و تعدادی if عدد هایی که کاربر وارد می‌کند را کنترل می کنیم که اگرکاربر اعداد خارج از محدوده واردکرد به او اخطار می دهد .

نکته مهم دیگر این است که دو ویژگی که تعداد ساعات هستند همانطورکه در قسمت های قبل آن ها را نرمالایز کردیم اینجا هم پس ازگرفتن ازکاربر آن ها را تقسیم بر ۱۰۰ میکنم و آن را ذخیره میکنیم.

نکته مهم دیگر این بخش این است؛ همان طورکه در صورت سوال گفته شده باید چه کرد که نمره حتما بین صفر تا بیست باشد در این جا باید به علل این موضوع توجه شود : یکی از علل این که نمره تولید شده در خارج از این بازه باشد این است که عدد های که کاربر به عنوان ورودی واردکرده به اشتباه در خارج از بازه تعیین شده بود و یا حتی عدد منفی واردکرده است که این مشکل را با دو تابع try و catch و چند if حل کردیم که اگر عدد وارد شده خارج از بازه تعریف شده بود اخطار دهد . پس اگر فرض کنیم که این که کاربر اطلاعات را درست واردکرده است نمره تخمین زده شده همیشه با وجود اینکه تمام ویژگی ها هم صفر وارد شود؛ حداقل نمره ایی میدهدکه این حداقل نمره به دلیل این است که یکی از وزن ها که به آن bias گفته می‌شود همیشه ۱است پس همیشه حداقل نمره ایی داریم. پس نمره‌ی کمتر از صفر نداریم ولی ممکن است نمره ی پیش بینی ما بیشتر از ۲۰ باشد که این مشکل را با گذاشتین یک if که چک میکندکه اگر نمره بیشتر از ۲۰ شده همان ۲۰ را برگرداند.کد این مرحله در صفحه‌ی بعد قابل مشاهد است.

```cpp
void predict(std::vector<double> w)
{
    std::vector<double> x(7,0);
    bool flag{ true };
    std::string error{ "your number must be Between 0 and 1 " };
    std::string error1{ "your number must be positive " };

    x[0] = 1;

    try{
        std::cout << "please enter your mark factors" << std::endl;
        std::cout << "mark of Attention in the class (Between 0 and 1) : ";
        std::cin >> x[1];
        if (x[1] <= 1 && x[1] >= 0)
            flag = true;
        else
            throw std::invalid_argument(error);
        std::cout << std::endl << "mark of Attention in the TA class (Between 0 and 1) : ";

        std::cin >> x[2];
        if (x[2] <= 1 && x[2] >= 0)
            flag = true;
        else
            throw std::invalid_argument(error);

        std::cout << std::endl << "Hours of coding and practicing in the week : ";
        std::cin >> x[3];
        if (x[3] >= 0)
            flag = true;
        else
            throw std::invalid_argument(error1);
        x[3] = x[3] / 100.;

        std::cout << std::endl << "Hours of studing and reading books in the week : ";
        std::cin >> x[4];
```

```
    if (x[4] >= 0)
        flag = true;
    else
        throw std::invalid_argument(error1);
    x[4] = x[4] / 100.;

    std::cout << std::endl << "mark of Previous background (Between 0 and 1) : ";
    std::cin >> x[5];
    if (x[5] <= 1 && x[5] >= 0)
        flag = true;
    else
        throw std::invalid_argument(error);

    std::cout << std::endl << "mark of Talent(Between 0 and 1) : ";
    std::cin >> x[6];
    if (x[6] <= 1 && x[6] >= 0)
        flag = true;
    else
        throw std::invalid_argument(error);
}
catch (std::invalid_argument& e)
{
    std::cerr << e.what()<<std::endl;
    flag = false;
}
if (flag)
{
    if (grade(w, x) > 20)
        std::cout << "your grade is : 20 " << std::endl;
    else
        std::cout << "your grade is : " << grade(w, x) << std::endl;
}
```

توجه: من ابتدا این برنامه را در visual studio 2019 نوشتم و عکس های کد نیز متعلق به همین برنامه است و در آخرکار کد را در visual studio code به وسیله‌ي docker اجرا کردم و نتایج درست بود.

```
root@1e47b9fb36ff:/usr/src/app# ./main
RUNNING TESTS ...
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from APHW1Test
[ RUN      ] APHW1Test.getDataFunctionTest
getData Test here!
[       OK ] APHW1Test.getDataFunctionTest (1 ms)
[ RUN      ] APHW1Test.gradeFunctionTest
grade Test here!
[       OK ] APHW1Test.gradeFunctionTest (1 ms)
[ RUN      ] APHW1Test.costFunctionTest
CostFunction Test here!
[       OK ] APHW1Test.costFunctionTest (3 ms)
[ RUN      ] APHW1Test.trainFunctionTest
Weights...
5.48528 ,4.46104 ,3.60295 ,2.61499 ,1.82324 ,2.32825 ,2.19782 ,
[       OK ] APHW1Test.trainFunctionTest (963 ms)
[----------] 4 tests from APHW1Test (973 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (975 ms total)
[  PASSED  ] 4 tests.
<<<SUCCESS>>>
root@1e47b9fb36ff:/usr/src/app#
```