در ابتدا شماره دانشجویی به عنوان دانه در کد قرار داده شد تا ستون‌های متناظر با کد دانشجویی انتخاب و عملیات پاک‌سازی و تحلیل و آنالیز داده‌ها تنها بر روی ستون‌های مربوطه قرار گیرد. در این بخش تغییراتی همچون نگاشت ستون به اعداد انجام شد تا بتوان راحت‌تر از ابزارهای تحلیل داده همچون پاندا و نامپای بهره برد. ستون‌های منتسب شده به بنده به شرح زیر بوده است:

```
['b', 'd', 'f', 'j', 'm', 'n', 'o', 't', 'w', 'x', 'y']
[1, 3, 5, 9, 12, 13, 14, 19, 22, 23, 24]
```

که این ستون‌ها درواقع ستون‌های زیر در فایل داده بوده‌اند:

	Customer Type	Type of Travel	Departure/Arrival time convenient	Food and drink	Online boarding	Seat comfort	Checkin service	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
0	Loyal Customer	Personal Travel	4	5	3	5	4	25	18.0	neutral or dissatisfied
1	disloyal Customer	Business travel	2	1	3	1	1	1	6.0	neutral or dissatisfied
2	Loyal Customer	Business travel	2	5	5	5	4	0	0.0	satisfied
3	Loyal Customer	Business travel	5	2	2	2	1	11	9.0	neutral or dissatisfied
4	Loyal Customer	Business travel	3	4	5	5	3	0	0.0	satisfied
5	Loyal Customer	Personal Travel	4	1	2	1	4	0	0.0	neutral or dissatisfied
6	Loyal Customer	Personal Travel	4	2	2	2	3	9	23.0	neutral or dissatisfied
7	Loyal Customer	Business travel	3	5	5	5	4	4	0.0	satisfied
8	Loyal Customer	Business travel	2	4	3	3	4	0	0.0	neutral or dissatisfied
9	disloyal Customer	Business travel	3	2	3	3	4	0	0.0	neutral or dissatisfied

پس از بارگذاری داده‌ها در کد با استفاده از کتابخانه پاندا، تحلیل‌هایی بر روی آن انجام دادم تا شناسایی و کار کردن با این مجموعه راحت‌تر شود. بدین منظور، در ابتدا نوع داده‌های هر ستون، تعداد سطر و ستون، بررسی وجود درایه‌هایی که مقادیری ندارند و در نهایت تعداد تکرار هر یک از درایه‌ها را با استفاده از توابع زیر استخراج کردم:

```
# Check the data types of the columns
print(df.dtypes, end='\n\n')

# Check the number of rows and columns in the dataset
print(df.shape, end='\n\n')


# Check the number of missing values in each column
print(df.isnull().sum(), end='\n\n')


# Check the number of unique values in each column
print(df.nunique(), end='\n\n')
```

در مراحل بعد، پاک‌سازی داده‌ها انجام شد که این مراحل به شرح زیر هستند:

۱. حذف داده‌های تکراری

می‌دانیم داده‌های تکراری می‌توانند منجر به نمایش غیرواقعی و نادرستی از داده‌ها شوند؛ همچنین داده‌های تکراری منابع را بی‌هدف هدر می‌دهند و کارکرد را کم می‌کنند. بنابراین به دلایل ذکر شده و با استفاده از تابع زیر، ردیف‌های تکراری حذف شدند. هرچند بر روی مجموعه داده‌ای که متناظر با کد بنده بوده است هیچ تکراری وجود نداشت که خروجی زیر بر این امر صحه می‌گذارد:

```
print(df.shape)

# Handle duplicate values
df.drop_duplicates(inplace=True)

print(df.shape)
```

✓ 0.0s

```
(103904, 25)
(103904, 25)
```

۲. حذف ردیف‌هایی که شامل داده نیستند

داده‌های تعریف نشده یا تهی همواره در یک مجموعه داده می‌توانند باعث در خطر انداختن پیوستگی داده‌ها شوند؛ همچنین اکثر تحلیل‌های آماری با این فرض انجام می‌شوند که داده‌ها به صورت کامل گردآوری شده‌اند. از همه مهم‌تر می‌دانیم سر و کار داشتن با داده‌هایی که مقادیر مشخصی ندارند می‌تواند باعث اضافه کردن پیچیدگی به سیستم تحلیل‌گر شود. پس با استفاده از توابع زیر از کتابخانه پاندا، ابتدا به سرشماری داده‌های تعریف نشده رسیدیم و پس از آن این ردیف‌ها از مجموعه داده حذف شدند:

```python
# missing data
missing_values_count = df.isnull().sum()
print(missing_values_count)

# eliminate rows with missing values
print(df.shape)
df.dropna(inplace=True)
print(df.shape)
```

تنها برای ستون تأخیر رسیدن، ۳۱۰ مقدار تهی وجود داشت که حـذف ردیف‌هـای متناظر با آن انجام شد:

```
Unnamed: 0                            0
id                                    0
Gender                                0
Customer Type                         0
Age                                   0
Type of Travel                        0
Class                                 0
Flight Distance                       0
Inflight wifi service                 0
Departure/Arrival time convenient     0
Ease of Online booking                0
Gate location                         0
Food and drink                        0
Online boarding                       0
Seat comfort                          0
Inflight entertainment                0
On-board service                      0
Leg room service                      0
Baggage handling                      0
Checkin service                       0
Inflight service                      0
Cleanliness                           0
Departure Delay in Minutes            0
Arrival Delay in Minutes            310
satisfaction                          0
dtype: int64
(103904, 25)
(103594, 25)
```

۳. فیلتر ستون‌ها متناظر با کد دانشجویی، حذف و حفظ برخی ستون‌ها

این بخش به محدودسازی ستون‌ها که در ابتدای کد نیز قابل انجام بود می‌پردازد. اما به دلیل اهمیت ذکر برخی نقاط جزئی، در این بخش به بررسی آن می‌پردازیم. در قطعه کد زیر فیلتر ستون‌ها متناظر با کد دانشجویی و با احتساب ستون بلا استفاده که در داده وجود داشت و برای خوانایی بیشتر کد به شکل زیر انجام شد:

```
# Filter the columns based on indices of selected features the way usecols works
selected_features.append(0)

df = df.iloc[:, selected_features]
df.head(10)
```

حال دو ستون بلا فایده که در داده‌ها وجود داشتند و کمکی به تمرین شبکه عصبی نمی‌کنند نیز حذف شدند:

```
# drop useless columns
df.drop(columns=['Unnamed: 0'], inplace=True)
df.drop(columns=['id'], inplace=True)

df.head(10)
```

خروجی در این بخش به شکل زیر است:

	Customer Type	Type of Travel	Departure/Arrival time convenient	Food and drink	Online boarding	Seat comfort	Checkin service	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
0	Loyal Customer	Personal Travel	4	5	3	5	4	25	18.0	neutral or dissatisfied
1	disloyal Customer	Business travel	2	1	3	1	1	1	6.0	neutral or dissatisfied
2	Loyal Customer	Business travel	2	5	5	5	4	0	0.0	satisfied
	Loyal	Business								neutral or

۴. تبدیل انواع داده

می‌دانیم در این پروژه در نهایت باید تحلیلی بر روی داده‌ها انجام بگیرد و یک مدلی طراحی شود تا بتواند برای داده‌هایی که برچسبی ندارند هم رضایت مسافران را با دقت خوبی پیش‌بینی کند. اما منظور از تبدیل نوع داده چیست و چرا؟ به صورت خلاصه ابزارهای آماری اکثراً الزامی برای فراهم نمودن فرمت و شکل خاصی از داده‌ها می‌کنند و در اینجا نیز ما برای آموزش شبکه عصبی و تنظیم وزن‌ها، نیاز داریم تا داده‌ها در فرمت خاصی طبقه‌بندی و تمیز شده باشند تا پردازش روی آن‌ها به راحتی صورت گیرد. همچنین در اینجا سعی شد تبدیل داده‌ها به داده‌های عددی صورت گیرد؛ چرا که می‌دانیم در کامپیوتر تحلیل و پردازش داده‌های عددی بسیار راحت‌تر از داده‌های رشته‌ای انجام می‌گیرد؛ بنابراین ما با تبدیل نوع داده‌ها را استاندارد کردیم. ستون‌هایی که نیاز به تبدیل داشته‌اند، نوع مشتری، نوع سفر و رضایت‌مندی مسافران بوده که از طبقه‌بندی رشته‌ای به طبقه‌بندی عددی تبدیل شدند:

```
df['Type of Travel'].replace({'Personal Travel': 0, 'Business travel': 1}, inplace=True)
df.head(10)
```

```
df['satisfaction'] = np.where(df['satisfaction'] == 'satisfied', 1, 0)
df.head(10)
```

```
df['Customer Type'].replace({'Loyal Customer': 0, 'disloyal Customer': 1}, inplace=True)
df.head(10)
```

در نهایت خروجی به شکل زیر شد که همگی در نوع عددی و یکپارچه هستند:

	Customer Type	Type of Travel	Departure/Arrival time convenient	Food and drink	Online boarding	Seat comfort	Checkin service	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
0	0	0	4	5	3	5	4	25	18.0	0
1	1	1	2	1	3	1	1	1	6.0	0
2	0	1	2	5	5	5	4	0	0.0	1
3	0	1	5	2	2	2	1	11	9.0	0
4	0	1	3	4	5	5	3	0	0.0	1
5	0	0	4	1	2	1	4	0	0.0	0
6	0	0	4	2	2	2	3	9	23.0	0
7	0	1	3	5	5	5	4	4	0.0	1
8	0	1	2	4	3	3	4	0	0.0	0
9	1	1	3	2	3	3	4	0	0.0	0

۵. استانداردسازی، نرمال‌سازی و مقیاس‌پذیری

پس از پاک‌سازی داده‌ها، حال یک نگاه کلی به آن با استفاده از توابع توصیف‌کننده داده‌ها در پاندا به صورت زیر است:

	Customer Type	Type of Travel	Departure/Arrival time convenient	Food and drink	Online boarding	Seat comfort	Checkin service	Departure Delay in Minutes	Arrival Delay in Minutes	satisfac
count	103594.000000	103594.000000	103594.000000	103594.000000	103594.000000	103594.000000	103594.000000	103594.000000	103594.000000	103594.00(
mean	0.182752	0.689857	3.060081	3.202126	3.250497	3.439765	3.304323	14.747939	15.178678	0.433
std	0.386465	0.462554	1.525233	1.329401	1.349433	1.318896	1.265396	38.116737	38.698682	0.495
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
25%	0.000000	0.000000	2.000000	2.000000	2.000000	2.000000	3.000000	0.000000	0.000000	0.00(
50%	0.000000	1.000000	3.000000	3.000000	3.000000	4.000000	3.000000	0.000000	0.000000	0.00(
75%	0.000000	1.000000	4.000000	4.000000	4.000000	5.000000	4.000000	12.000000	13.000000	1.00(
max	1.000000	1.000000	5.000000	5.000000	5.000000	5.000000	5.000000	1592.000000	1584.000000	1.00(

با نگاه کلی، می‌توان فهمید که اکثر ستون‌ها، به جز تأخیر رسیدن و تأخیر خروج دارای محدوده خاصی هستند یا مقادیر طبقه‌بندی شده‌ای را دارند؛ اما برای این دو ستون، مشخص است که داده‌ها محدوده خاصی ندارند و می‌توانند مقادیر بسیار پرتی را نیز اختیار کنند. برای کنترل نمودن این وضعیت و دقت بالاتر در داده‌ها، همچنین کاهش انحراف از معیار، داده‌های پرت را به روش زیر از مجموعه داده حذف می‌کنیم:

```
# deal with outliers
df = df[df['Departure Delay in Minutes'] < 100]
df = df[df['Arrival Delay in Minutes'] < 100]

df.describe()
```

✓ 0.1s Python

	Customer Type	Type of Travel	Departure/Arrival time convenient	Food and drink	Online boarding	Seat comfort	Checkin service	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
count	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000
mean	0.182360	0.689379	3.060284	3.208661	3.253078	3.445359	3.307066	8.922623	9.157505	0.436432
std	0.386143	0.462750	1.526528	1.328461	1.349680	1.318105	1.263039	17.596357	17.630477	0.495945
min	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	2.000000	2.000000	2.000000	2.000000	3.000000	0.000000	0.000000	0.000000
50%	0.000000	1.000000	3.000000	3.000000	3.000000	4.000000	3.000000	0.000000	0.000000	0.000000
75%	0.000000	1.000000	4.000000	4.000000	4.000000	5.000000	4.000000	9.000000	10.000000	1.000000
max	1.000000	1.000000	5.000000	5.000000	5.000000	5.000000	5.000000	99.000000	99.000000	1.000000

همانطور که مشخص است، توانستیم داده‌ها را یکنواخت‌تر کنیم و از تحلیل داده‌های بسیار پرت که کم‌اهمیت‌تر هستند خودداری کنیم.

پس از اینکه داده‌های پرت را حذف کردیم، حال یک نوع از مقیاس‌پذیری‌ها و یک نوع از استانداردسازی‌ها پیاده‌سازی شدند:

```python
# Perform normalization and scaling

# # Min-Max Scaling
ddm = df['Departure Delay in Minutes']
adm = df['Arrival Delay in Minutes']
df['Departure Delay in Minutes'] = (ddm - ddm.min()) / (ddm.max() - ddm.min())
df['Arrival Delay in Minutes'] = (adm - adm.min()) / (adm.max() - adm.min())

# Z-score
# ddm = df['Departure Delay in Minutes']
# adm = df['Arrival Delay in Minutes']
# df['Departure Delay in Minutes'] = (ddm - ddm.mean()) / ddm.std()
# df['Arrival Delay in Minutes'] = (adm - adm.mean()) / adm.std()


df.describe()
```

✓ 0.0s Python

	Customer Type	Type of Travel	Departure/Arrival time convenient	Food and drink	Online boarding	Seat comfort	Checkin service	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
count	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000	99578.000000
mean	0.182360	0.689379	3.060284	3.208661	3.253078	3.445359	3.307066	0.090128	0.092500	0.436432
std	0.386143	0.462750	1.526528	1.328461	1.349680	1.318105	1.263039	0.177741	0.178086	0.495945
min	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	2.000000	2.000000	2.000000	2.000000	3.000000	0.000000	0.000000	0.000000
50%	0.000000	1.000000	3.000000	3.000000	3.000000	4.000000	3.000000	0.000000	0.000000	0.000000
75%	0.000000	1.000000	4.000000	4.000000	4.000000	5.000000	4.000000	0.090909	0.101010	1.000000
max	1.000000	1.000000	5.000000	5.000000	5.000000	5.000000	5.000000	1.000000	1.000000	1.000000

مقیاس‌پذیری min-max که در اینجا پیاده‌سازی شده، وظیفه نرمالیزه کردن محدوده featureها را بر عهده دارد. این فرمول داده‌ها را به محدوده بین صفر تا یک مقایس می‌کند. لازم به ذکر است بدیهی است که این مقیاس‌پذیری، همبستگی و ارتباط بین داده‌ها (در اینجا فاصله از هم) را حفظ می‌کند و حتی می‌تواند مینیمم و ماکزیمم را نیز به شدت تحت تأثیر قرار دهد!

بخش دوم کد که به حالت کامنت در آمده، استانداردسازی Z-score است که نقطه داده فعلی را نسبت به میانگین بر حسب انحراف معیار توصیف می‌کند. من در اینجا به این دلیل که شاید دو عمل بر یکدیگر تأثیر منفی بگذارند و دقت خروجی را تحت تأثیر قرار بدهند، تنها از یکی از آن‌ها را استفاده کرده‌ام و در نهایت نیز دقت خوبی را در خروجی شبکه کسب کرده‌ام.

در ادامه و بخش بعدی جدولی تشکیل شده از جایگشت و عملیات‌های متفاوت را ارائه خواهم داد تا دقت شبکه در هر یک از این حالات سنجیده شود.

۵. تمرین مدل و تشکیل شبکه عصبی

همانطور که پیش‌تر اشاره شد، در این پروژه برای پیش‌بینی رضایت مسافران آزمایشی یا مسافرانی که برچسب مشخصی ندارند، از روش شبکه عصبی برای پیش‌بینی آن استفاده شد. به صورت خلاصه به تعریف شبکه عصبی و اجزای آن می‌پردازیم.

شبکه عصبی درواقع یک مدل محاسباتی الگو گرفته شده از شبکه عصبی بیولوژیکی مغز انسان و روش پردازش اطلاعات آن است. ساختار کلی شبکه عصبی به صورت لایه لایه است که هر لایه شامل تعدادی نورون می‌باشد. این لایه‌ها شامل لایه‌ی ورودی، یک یا چند لایه مخفی و لایه خروجی است و هر یک از نورون‌های هر لایه، با وزن‌هایی به نورون‌های لایه بعد وصل هستند که می‌تواند یک شبکه کاملاً متصل یا Fully Connected را بسازد. وزن‌ها و بایاس‌ها درواقع بخش اصلی تمرین یک شبکه عصبی هستند و پارامترهایی هستند که در عین آموزش یاد گرفته می‌شوند. وزن، قدرت اتصال میان دو نورون را مشخص می‌کند و بایاس برای تطبیق بهتر شبکه بر داده‌های ورودی اعمال می‌شود. در ساختاری که برای این پروژه در نظر گرفته شده، ما داده‌های ورودی با اندازه ۸ ویژگی را داریم که بنابراین به ۸ نورون برای لایه ورودی نیاز داریم که این ۸ وِیژگی در بخش ابتدایی نمایش داده شده بود و به صورت زیر می‌باشد:

	Customer Type	Type of Travel	Departure/Arrival time convenient	Food and drink	Online boarding	Seat comfort	Checkin service	Departure Delay in Minutes	Arrival Delay in Minutes	satisfaction
0	0	0	4	5	3	5	4	25	18.0	0
1	1	1	2	1	3	1	1	1	6.0	0
2	0	1	2	5	5	5	4	0	0.0	1
3	0	1	5	2	2	2	1	11	9.0	0
4	0	1	3	4	5	5	3	0	0.0	1
5	0	0	4	1	2	1	4	0	0.0	0
6	0	0	4	2	2	2	3	9	23.0	0
7	0	1	3	5	5	5	4	4	0.0	1
8	0	1	2	4	3	3	4	0	0.0	0
9	1	1	3	2	3	3	4	0	0.0	0

پس ما از اطلاعات این ۸ ستون (بدون ستون برچسب یا همان رضایت‌مندی) به تعلیم شبکه می‌پردازیم.

اجزای اصلی شبکه در کد تشکیل شده از بخش‌های تابع آغازین، تابع انتشار به سمت جلو، انتشار به سمت عقب، بروزرسانی پارامترها، پیش‌بینی، تعلیم، تابع هزینه است.

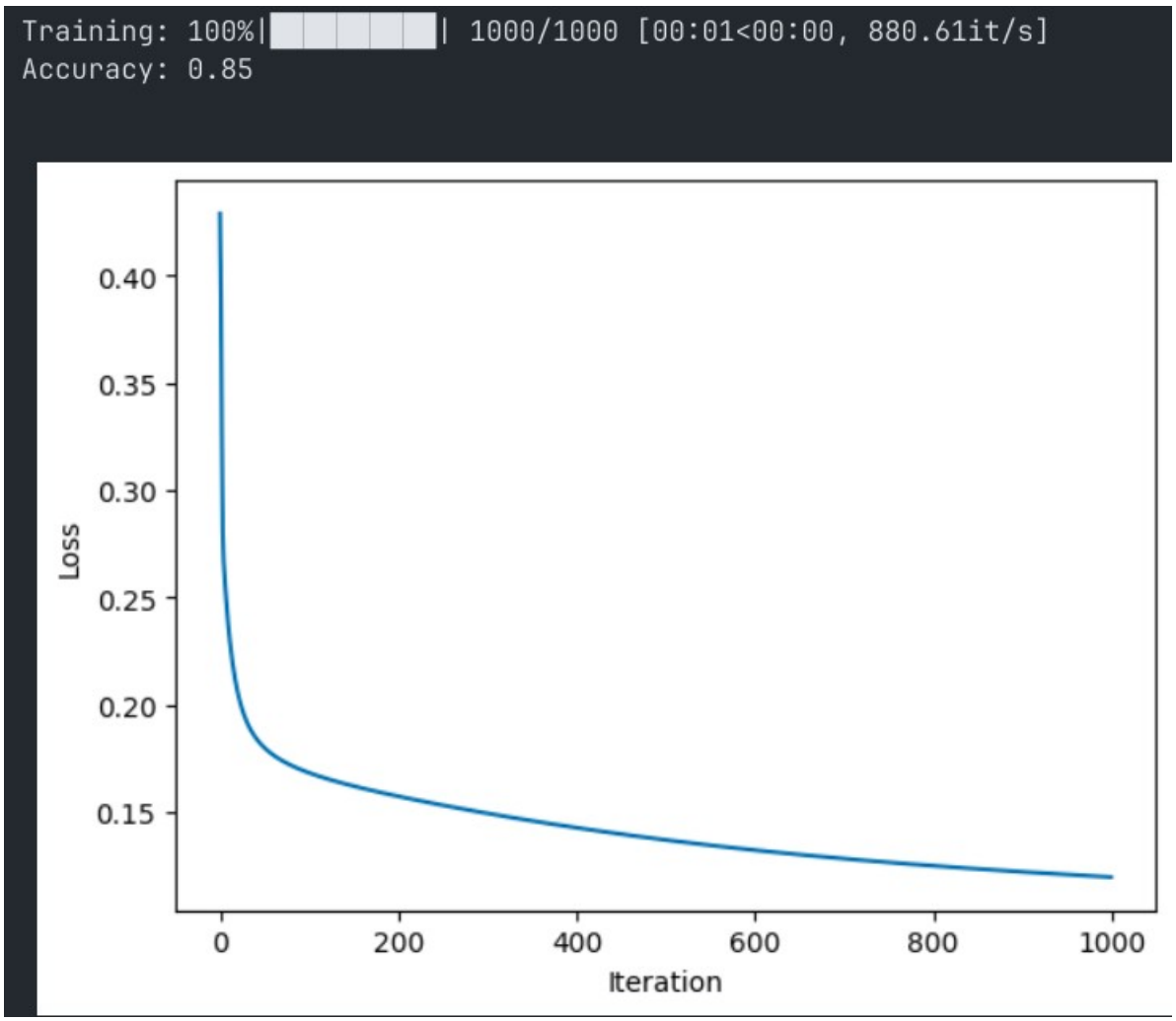
در تابع آغازین، پس از اینکه از کلاس شبکه عصبی شی‌ای ساخته شد، باید تعداد نورون‌های ورودی، خروجی و تک لایه مخفی (من برای شبکه عصبی خود تنها یک لایه را در نظر گرفتم) مشخص شود تا بتوان ماتریس‌های وزن و بایاس را متناظر با آن تنظیم نمود. ماتریس‌های وزن اولی در میان لایه ورودی و لایه مخفی اول و ماتریس‌های وزن دومی در میان لایه مخفی و لایه خروجی قرار می‌گیرند.

ماتریس وزن اولی $h*8$ و ماتریس وزن دومی $2*h$ می‌باشد. می‌دانیم سایز خروجی باید ۲ باشد چرا که مسافران در دو طبقه خنثی، ناراضی یا راضی قرار می‌گیرند که این دو را با استفاده از یک مقدار آستانه از هم تمیز می‌دهیم تا خروجی نهایی مشخص شود. بایاس‌ها نیز دو بردار به اندازه تعداد نورون‌های لایه پنهان و لایه خروجی هستند که به حاصل‌ضرب وزن‌ها و خروجی لایه‌های قبلی اضافه می‌شوند.

پس از مقداردهی اولیه وزن‌ها و بایاس‌ها، حال بخش انتشار به سمت جلو را می‌سازیم. انتشار به سمت جلو درواقع همان ضرب نقطه‌ای میان ورودی و ماتریس وزن اول که به آن بایاس لایه اول اضافه می‌شود و ضرب نقطه‌ای میان خروجی این لایه و ماتریس وزن دوم که بایاس لایه دوم به آن اضافه می‌شود است. البته لازم به ذکر است تابع فعال‌سازی در نظر گرفته شده برای خروجی هر لایه این شبکه Sigmoid است.

تابع انتشار به سمت عقب نیز در بخش آموزش برای محاسبه مشتق و محاسبات مربوط به گرادیان کاهشی پیاده‌سازی شده است که در تابع بروزرسانی پارامترها، ما وزن‌ها را در هر مرحله بروزرسانی می‌کنیم تا شبکه تعلیم ببیند و وزن و بایاس خود را متناسب با داده‌ها تنظیم کند.

حال که اجزای اصلی شبکه ساخته شد، نوبت به ایجاد ساختار آموزش آن می‌رسد؛ تابع آموزش در کد وظیفه دارد تا داده‌های آموزشی را به تعداد iteration بار یا همان تعداد epoch ها از شبکه عبور دهد تا وزن‌ها تنظیم شوند و بتوانند داده‌های آزمایشی را درست پیش‌بینی کنند. در این تابع علاوه بر انتشار به سمت جلو، انتشار به سمت عقب و بروزرسانی پارامترها که همان وزن و بایاس هستند نیز صورت می‌گیرد. از تعداد ۲۰۰۰ داده رندومی که در مستند ذکر شد، ۸۰ درصد آن برای آموزش و ۲۰ درصد آن برای آزمایش در نظر گرفته شدند. بنابراین ابتدا ۱۶۰۰ ردیف با برچسب را به تابع آموزش می‌دهیم تا شبکه تعلیم شود، سپس ۴۰۰ داده بدون برچسب را از شبکه عبور می‌دهیم تا برچسب‌های آن را تعیین کند و در نهایت دقت شبکه را با انطباق دادن ۴۰۰ برچسب بدست آمده از شبکه با برچسب‌های واقعی تعیین می‌کنیم:



همانطور که در شکل بالا مشخص شده است، شبکه با دقت ۸۵ درصد داده‌های آزمایشی را به درستی پیش‌بینی کرده است.

در جداول زیر، دقت به ازای استفاده از نرمال‌سازی سازی و مقادیر متفاوت نرخ یادگیری مشخص شده است (البته بدیهی است به دلیل انتخاب ردیف‌ها به صورت رندوم، شاید تحلیل‌ها دقیق نباشد):

با ۶۴ نورون و Min-Max Scaling:

Learning Rate	0.01	0.1	0.5	1
Accuracy	0.78	0.81	0.87	0.82

با نرخ یادگیری ۰.۵ و Min-Max Scaling:

Hidden Layer Neurons	16	32	64	128
Accuracy	0.85	0.81	0.87	0.85

با نرخ یادگیری ۰.۵ و ۶۴ نورون:

Z-score	+		+
Min-Max		+	+
Accuracy	0.81	0.87	0.78

خروجی نهایی به همراه نمودار هزینه با نرخ یادگیری ۰.۵، ۶۴ نورون لایه پنهان و Min-Max Scaling: