دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکدهٔ مهندسی برق

درس یادگیری ماشین

پاسخ مینی پروژه۱

نام و نام خانوادگی	مهدی خدابنده‌لو
شمارهٔ دانشجویی	۴۰۱۰۵۳۱۴
تاریخ	بهار ۱۴۰۳

فهرست مطالب

۱ سوال اول ۵
۱.۱ فرآیند آموزش و ارزیابی یک مدل طبقه بند خطی را به صورت دیاگرامی بلوکی نمایش دهید و در مورد اجزای مختلف این دیاگرام بلوکی توضیحاتی بنویسید. تغییر نوع طبقه بندی از حالت دوکلاسه به چندکلاسه در کدام قسمت از این دیاگرام بلوکی تغییراتی ایجاد می کند؟ توضیح دهید. ۵
۲.۱ با استفاده از یکsklearn.datasets، دیتاست با ۱۰۰۰نمونه، ۴کلاس و ۳ویژگی تولید کنید و آن را به صورتی مناسب نمایش دهید. آیا دیتاستی که تولید کردید چالش برانگیز است؟ چرا؟ به چه طریقی می توانید دیتاست تولیدشده خود را چالش برانگیزتر و سخت تر کنید؟ ۵
۳.۱ با استفاده از حداقل دو طبقه‌بند خطی آماده‌ی پایتون و در نظر گرفتن فراپارامترهای مناسب چهار کلاس موجود در دیتاست قبلی را از هم تفکیک کنید. ضمن روند توضیح فراپارامترها نتیجه دقت آموزش و ارزیابی را نشان دهید. برای بهبود نتایج از چه روش‌هایی استفاده کردید؟ ۶

۲ سوال دوم ۸
۱.۲ با مراجعه به صفحهٔ دیتاست CWRU باBearing یک دیتاست مربوط به حوزهٔ تشخیص عیب آشنا شوید. باجستجوی آن در اینترنت و مقالات، توضیحاتی از اهداف، ویژگی ها و حالت های مختلف این دیتاست ارائه کنید. ۸
۲.۲ برای تشکیل دیتاست مراحل زیر را انجام دهید: ۸
۳.۲ بدون استفاده از کتابخانه های آمادهٔ پایتون، مدل طبقه بند، تابع اتلاف و الگوریتم یادگیری و ارزیابی را کدنویسی کنیدتا دو کلاس موجود در دیتاست به خوبی از یکدیگر تفکیک شوند. نمودار تابع اتلاف را رسم کنید و نتیجهٔ ارزیابی رویداده های تست را با حداقل ۲شاخصه محاسبه کنید. نمودار تابع اتلاف را تحلیل کنید. آیا می توان از روی نمودارتابع اتلاف و قبل از مرحلهٔ ارزیابی با قطعیت در مورد عمل کرد مدل نظر داد؟ چرا و اگر نمی توان، راه حل چیست؟ ۱۲

۳ سوال سوم ۱۴
۱.۳ ابتدا هیت مپ ماتریس همبستگی و هیستوگرام پراکندگی ویژگی ها را رسم و تحلیل کنید ۱۴

فهرست تصاویر

۱ داده های تولید شده با class_sep=۱ . . . . . ۶
۲ داده های تولید شده با class_sep=۰.۵ . . . . . ۷
۳ خروجی تابع Loss در هر مرحله ۱۴
۴ قدر مطلق وزن ها ۱۵
۵ هیت مپ ۱۶
۶ هیستوگرام ۱۷

فهرست جداول

فهرست برنامه‌ها

۱ (Python) data Generate ۵
۲ (Python) split test and Train ۶
۳ accuracy(Python) Calculate ۷
۴ (Python) ۸
۵ (Python) ۹
۶ (Python) ۱۰
۷ (Python) ۱۰
۸ (Python) ۱۱
۹ (Python) ۱۲
۱۰ (Python) ۱۳
۱۱ (Python) ۱۳
۱۲ (Python) ۱۴

Notebook Colab
github

۱ سوال اول

۱.۱ فرآیند آموزش و ارزیابی یک مدل طبقه بند خطی را به صورت دیاگرامی بلوکی نمایش دهید و در مورد اجزای مختلف این دیاگرام بلوکی توضیحاتی بنویسید. تغییر نوع طبقه بندی از حالت دوکلاسه به چندکلاسه در کدام قسمت از این دیاگرام بلوکی تغییراتی ایجاد می کند؟ توضیح دهید.

۲.۱ با استفاده از یک sklearn.datasets، دیتاست با ۱۰۰۰نمونه، ۴کلاس و ۳ویژگی تولید کنید و آن را به صورتی مناسب نمایش دهید. آیا دیتاستی که تولید کردید چالش برانگیز است؟ چرا؟ به چه طریقی می توانید دیتاست تولیدشده خود را چالش برانگیزتر و سخت تر کنید؟

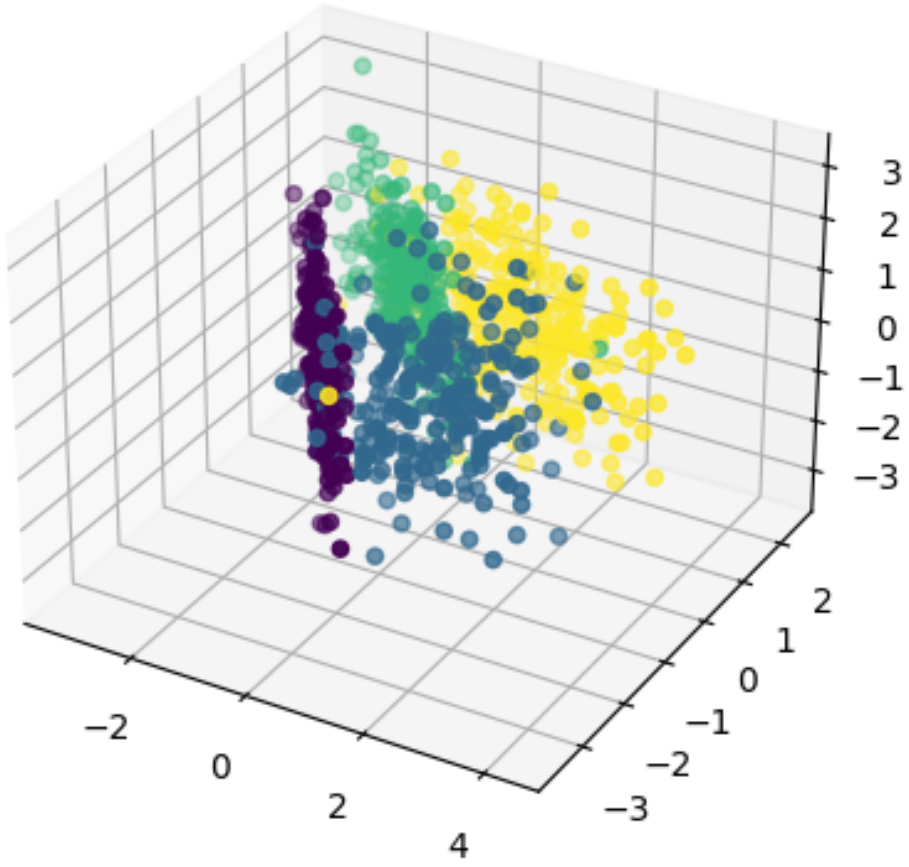
همان طور که در Code۱ قابل مشاهده است با استفاده از دستور make_classification نقطه را تولید کرده‌ایم. و سپس آن ها را در یک نمودار سه بعدی نمایش داده‌ایم. شکل۱ این نقاط را نمایش می‌دهد.

```
X,y = make_classification(n_samples=1000, n_features=3, n_classes=4,
        n_redundant=0, random_state=14,
        n_clusters_per_class=1, class_sep=1)

fig = plt.figure()
ax = fig.add_subplot(projection='3d')
ax.scatter(X[:,0], X[:,1], X[:,2], c=y)
```

Code 1: Generate data (Python)

همان طور که از شکل۱ پیداست دیتاستی که تولید شده زیاد چالش برانگیز نمی‌باشد و می‌توان داده ها را از هم تفکیک کرد. برای این که داده های تولید شده را چالش بر انگیز تر کنیم می‌توانیم کارهای مختلفی انجیم دهیم.

یکی از این کارها این است که فراپارامتر class_sep را کاهش دهیم. شکل۲ داده‌ها را در حالتی که این پارامتر 0.4 است نمایش می‌دهد. همان طور که مشخص است داده‌های کلاس‌های مختلف به هم نزدیک تر شه اند وتفکیک آن‌ها مشکل تر شده است.



شکل ۱: داده های تولید شده با class_sep=۱

۳.۱ با استفاده از حداقل دو طبقه‌بند خطی آماده‌ی پایتون و در نظر گرفتن فراپارامترهای مناسب چهار کلاس موجود در دیتاست قبلی را از هم تفکیک کنید. ضمن روند توضیح فراپارامترها نتیجه دقت آموزش و ارزیابی را نشان دهید. برای بهبود نتایج از چه روش‌هایی استفاده کردید؟

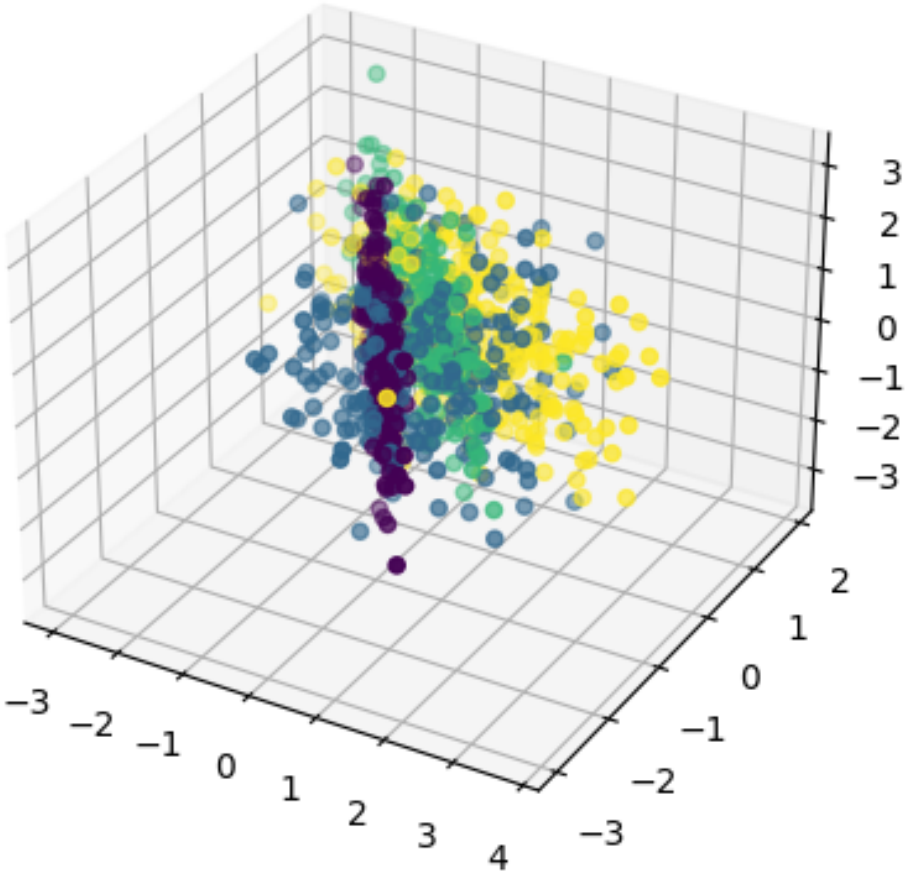
جدا کردن داده‌های آموزش و تست:

در Code۲ داده‌های آموزش و تست را به نسبت ۸۰ به ۲۰ تقسیم کرده‌ایم.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=14)
```

Code 2: Train and test split (Python)

Logistic_regression



شکل ۲: داده های تولید شده با class_sep=۰.۵

در Code۳ ابتدا تعداد تکرار برای آموزش را ۵۰ در نظر گرفته‌ایم و پس از آموزش، با استفاده از داده‌های تست دقت مدل را اندازه گیری کرده‌ایم. با مقادیری که یرای هر یک از وارامتر ها در نظر گرفته شده دقت برابر با ۸۹ درصد است.

```
logistic_regression_model = LogisticRegression(max_iter=50, random_state=14)
logistic_regression_model.fit(X=X_train, y=y_train)

y_hat = logistic_regression_model.predict(X_test)
y_hat_prob = logistic_regression_model.predict_proba(X_test)
accuracy = logistic_regression_model.score(X_test, y_test)
print('accuracy :', accuracy*100, '%')
```

Code 3: Calculate accuracy(Python)

برای بهبود نتایج باید تعداد تکرار برای آموزش را مناسب در نظر بگیریم. در صورتی که این پارامتر زیاد در نظر گرفته شده باشد باعث overfitting می‌شود، این پدیده باعث می‌شود که مدل روی داده‌های آموزش عملکرد خوبی داشته باشد ولی وقتی داده‌هایی غیر از داده‌های آموزش به مدل داده شود عملکرد آن خراب می‌شود. این پدیده مانند این است که یک دانش آموز به حای یاد گرفتن درس آن را حفظ کرده باشد. مسئله‌ی دیگری که در بهبود نتایج موثر است تعداد کافی داده‌های آموزش است، در صورتی مه تعداد داده‌های آموزش کم باشد مدل به خوبی آموزش نمی‌بیند.

۲ سوال دوم

۱.۲ با مراجعه به صفحهٔ دیتاست CWRU باBearing یک دیتاست مربوط به حوزهٔ تشخیص عیب آشنا شوید. باجستجوی آن در اینترنت و مقالات، توضیحاتی از اهداف، ویژگی ها و حالت های مختلف این دیتاست ارائه کنید.

برای تشخیص عیب از دو ویژگی drive end accelerometer data (که در کد x1 نامیده شده) و fan end accelerometer data (که در کد x2 نامیده شده) استفاده شده است.

۲.۲ برای تشکیل دیتاست مراحل زیر را انجام دهید:

الف: از هر کلاس M نمونه به طول N جدا کنید(M حد اقل ۱۰۰ و N حداقل ۲۰۰ باشد).یک ماتریس از داده های هر دو کلاس به همراه برچسب آن‌ها تشکیل دهید. می‌توانید پنجره‌ای به طول N در نظر بگیرید و در نهایت یک ماتریس MxN از داده‌های هر کلاس استخراج کنید.

ما در این جا M را برابر با ۵۰۰ و N را ۲۰۰ در نظر گرفته‌ایم. همان طور که از Code۴ مشخص است هر یک از ویژگی‌های کلاس‌های معیوب و سالم را با استفاده از دستور reshape به صورت مربعی درآورده ایم.

```
M = 500
N = 200
x1_normal = np.reshape(bearing_normal_dataset['X098_DE_time'][0:M*N], (M,N))
x2_normal = np.reshape(bearing_normal_dataset['X098_FE_time'][0:M*N], (M,N))
x1_fault = np.reshape(bearing_fault_dataset['X108_DE_time'][0:M*N], (M,N))
x2_fault = np.reshape(bearing_fault_dataset['X108_FE_time'][0:M*N], (M,N))
```

Code 4: (Python)

ب: در مورد اهمیت استخراج ویژگی در یادگیری ماشین توضیحاتی بنویسید.سپس، با استفاده از حداقل هشت عدد از روش های ذکر شده در جدول۱، ویژگی‌های دیتاست قسمت «۲-آ» را استخراج کنید و یک دیتاست جدید تشکیل دهید.

Code۵ تابعی که به منظور استخراج ویژگی نوشته شده را نشان می‌دهد. ورودی این تابع ویژگی‌های دیتاست خام می‌باشد و خروجی آن یک dictionary است که شامل ویژگی های استخراج شده از ویژگی های خام و خود ویژگی‌های خام و خود ویژگی‌های خام است.

این تابع تمامی ویژگی‌های ذکر شده در جدول را محاسبه می‌کند. نکته‌ی مهمی که در این تابع وحود دارد این است که به منظور جلوگیری از تقسیم شدن بر صفر، زمانی که مقسوم علیه صفر شود، به جای صفر مقدار اپسیلون قرار می‌گیرد.

```python
def feature_extract(x1,x2,ep):
  x1_n = np.expand_dims(x1, axis=2)
  x2_n = np.expand_dims(x2, axis=2)
  xc = np.concatenate((x1_n,x2_n), axis=2)
  xc_mean = np.mean(xc, axis=2)
  m = np.expand_dims(xc_mean, axis=2)
  m = np.concatenate((m,m), axis=2)

  feature = {'x1': x1,'x2': x2}
  feature['std']  = np.std(xc, axis=2)
  feature['p']    = np.amax(np.abs(xc), axis=2)
  temp = np.std(xc, axis=2)**3
  temp[temp == 0] = ep
  feature['ske']  = np.mean((xc-m)**3, axis=2) / temp
  temp = np.std(xc, axis=2)**4
  temp[temp == 0] = ep
  feature['kur']  = np.mean((xc-m)**4, axis=2) / temp
  feature['rms']  = np.sqrt(np.mean(xc**2, axis=2))
  temp = np.sqrt(np.mean(xc**2, axis=2))
  temp[temp == 0] = ep
  feature['cf']   = np.amax(np.abs(xc), axis=2) / temp
  feature['smr']  = np.mean(np.sqrt(np.abs(xc)), axis=2) ** 2
  temp = np.mean(np.sqrt(np.abs(xc)), axis=2) ** 2
  temp[temp == 0] = ep
  feature['clf']  = np.amax(np.abs(xc), axis=2) / temp
  temp = np.mean(np.abs(xc), axis=2)
  temp[temp == 0] = ep
  feature['sf']   = np.sqrt(np.mean(xc**2, axis=2)) / temp
  temp = np.mean(np.abs(xc), axis=2)
  temp[temp == 0] = ep
  feature['if1']  = np.amax(np.abs(xc), axis=2) / temp
  feature['if2']  = np.max(np.abs(xc), axis=2) / temp
  return feature
```

Code 5: (Python)

همان طور که از Code۶ مشخص است مقدار اپسیلون را $10 - 10$ در نظر گرفته‌ایم. پس از این مرحله داده‌های کلاس normal و fault را با هم ترکیب می‌کنیم. البته نیاز است که قبل از استفاده از داده‌ها آن‌ها را به خوبی بر بزنیم.

```
x_normal = feature_extract(x1_normal, x2_normal,1e-10)
x_fault = feature_extract(x1_fault, x2_fault,1e-10)

X = {'x1': np.concatenate((x_normal['x1'], x_fault['x1']), axis=0),
     'x2': np.concatenate((x_normal['x2'], x_fault['x2']), axis=0),
     'std': np.concatenate((x_normal['std'], x_fault['std']), axis=0),
     'p': np.concatenate((x_normal['p'], x_fault['p']), axis=0),
     'ske': np.concatenate((x_normal['ske'], x_fault['ske']), axis=0),
     'kur': np.concatenate((x_normal['kur'], x_fault['kur']), axis=0),
     'rms': np.concatenate((x_normal['rms'], x_fault['rms']), axis=0),
     'cf': np.concatenate((x_normal['cf'], x_fault['cf']), axis=0),
     'smr': np.concatenate((x_normal['smr'], x_fault['smr']), axis=0),
     'clf': np.concatenate((x_normal['clf'], x_fault['clf']), axis=0),
     'sf': np.concatenate((x_normal['sf'], x_fault['sf']), axis=0),
     'if1': np.concatenate((x_normal['if1'], x_fault['if1']), axis=0),
     'if2': np.concatenate((x_normal['if2'], x_fault['if2']), axis=0)}
y0 = np.zeros((M,N))#normal
y1 = np.ones((M,N))#fault
y = np.concatenate((y0,y1), axis=0)
```

Code 6: (Python)

ج: ضمن توضیح اهمیت فرآیند بر زدن، داده‌ها را در صورت امکان مخلوط کرده و با نسبت تقسیم دلخواه و معقول به دو بخش آموزش و ارزیابی تقسیم کنید.

در صوری که داده‌های کلاس‌های مختلف به خوبی مخلوط نشوند فرایند یادگیری به درستی انجام نمی‌شود. در Code۷ ابتدا داده‌ها را به شکل بردار درآورده یم و آن ها را به صورت یکسان مخلوط کرده‌ایم. پس از مخلوط کردن داده‌ها را به نسبت ۸۰ به ۲۰ به داده‌های آموزش و تست تقسیم کرده‌ایم.

```
data_shape = y.shape
for key,value in X.items():
  X[key] = X[key].flatten()
y = y.flatten()
np.random.seed(14)
```

```python
perm = np.random.permutation(len(y))
for key,value in X.items():
  X[key] = X[key][perm]
y = y[perm]
for key,value in X.items():
  X[key] = X[key].reshape(data_shape)
y = y.reshape(data_shape)
#train and test split
train_size = 0.8
train_a = round(0.8*M)
X_train = {}
X_test = {}
for key, value in X.items():
  X_train[key] = X[key][0:train_a,:]
  X_test[key] = X[key][train_a:,:]
y_train = y[0:train_a,:]
y_test = y[train_a:,:]
```

Code 7: (Python)

د: حداقل دو روش برای نرمال سازی داده ها را با ذکر اهمیت این فرآیند توضیح دهید و با استفاده از یکی از اینروش ها، داده ها را نرمال کنید. آیا از اطلاعات بخش »ارزیابی« در فرآیند نرمال سازی استفاده کردید؟ چرا؟

از آن جایی که رنج تغییرات ویژگی‌های مختلف متفاوت است ممکن است برخی از آنها که دامنه‌ی بزرگتری دارند اثر ویژگی‌هایی که دامنه‌ی کم تری دارند را خنثی کنند، از این رو بهتر است برای جلوگیری از این پدیده داده‌ها را نرمالیزه کنیم. Code۸ مربوط به نرمالیزه کردن داده‌های تست و آموزش به صورت جداگانه است.

```python
for key, value in X_train.items():
  X_train[key] =  (X_train[key] - np.min(X_train[key])) /...
      (np.max(X_train[key]) - np.min(X_train[key]))
for key, value in X_test.items():
  X_test[key] =  (X_test[key] - np.min(X_test[key])) /...
     (np.max(X_test[key]) - np.min(X_test[key]))
```

Code 8: (Python)

۳.۲ بدون استفاده از کتابخانه های آمادهٔ پایتون، مدل طبقه بند، تابع اتلاف و الگوریتم یادگیری و ارزیابی را کدنویسی کنیدتا دو کلاس موجود در دیتاست به خوبی از یکدیگر تفکیک شوند. نمودار تابع اتلاف را رسم کنید و نتیجهٔ ارزیابی رویداده های تست را با حداقل ۲شاخصه محاسبه کنید. نمودار تابع اتلاف را تحلیل کنید. آیا می توان از روی نمودارتابع اتلاف و قبل از مرحلهٔ ارزیابی با قطعیت در مورد عمل کرد مدل نظر داد؟ چرا و اگر نمی توان، راه حل چیست؟

همان طور که در Code۹ قابل مشاهده است به ترتیب توابع سیگموید، Logistic_regression، LOSS گرادیان، گرادیان نزولی و دقت پیاده‌سازی شده‌اند.

```
def sigmoid(x):
  return 1 / (1 + np.exp(-x))

def logestic_regression(x,w):
  u = 0
  for key,value in x.items():
    u += x[key] * w[key]
  h_hat = sigmoid(u)
  return h_hat

def bce(y, y_hat):
  ep = 1e-10
  loss = -(np.mean(y*np.log(y_hat + ep) + (1-y)*np.log(1-y_hat + ep)))
  return loss

def gradient(x, y, y_hat):
  grd = {}
  for key, value in x.items():
    grd[key] = np.sum(np.multiply(x[key],(y_hat-y)))
  return grd

def gradient_decent(w, eta, grads):
  for key, value in w.items():
    w[key] -= eta * grads[key]
  return w

def accuracy(y, y_hat):
```

```
  acc = np.sum(y == np.round(y_hat)) / y.size
  return acc
```

Code 9: (Python)

در Code۱۰ مقادیر اولیه‌ی وزن‌ها، تعداد تکرار برای آموزش و گام آموزش تعیین شده است.

```
X_train['bias'] = np.ones(X_train['x1'].shape)
np.random.seed(14)
w_initial = np.random.rand(len(X)+1,1)
w = {'bias': w_initial[0,0], 'x1': w_initial[1,0], 'x2': w_initial[2,0],
     'std': w_initial[3,0], 'p': w_initial[4,0], 'ske': w_initial[5,0],
     'kur': w_initial[6,0], 'rms': w_initial[7,0], 'cf': w_initial[8,0],
     'smr': w_initial[9,0], 'clf': w_initial[10,0], 'sf': w_initial[11,0],
     'if1': w_initial[12,0], 'if2': w_initial[13,0]}
initial_eta = 0.00005
eta = initial_eta
n_epochs = 1000
```

Code 10: (Python)

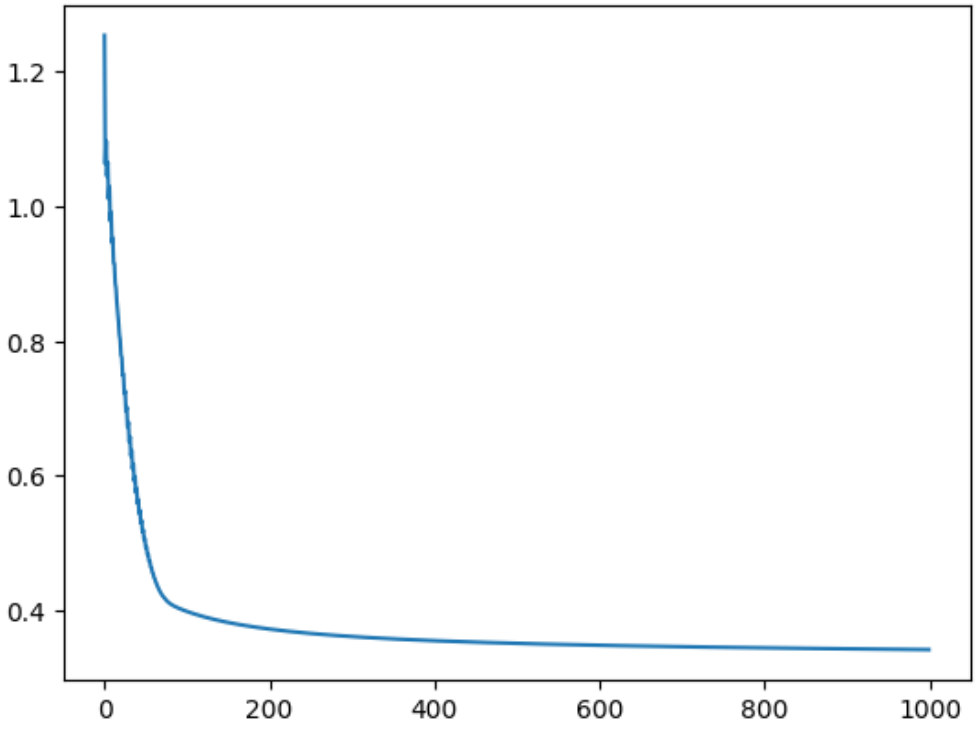
پس از مراحل بالا نوبت به آموزش مدل می‌رسد. Code۱۰ مربوط به آموزش مدل است که در هر اجرا وزن‌ها در آن آپدیت می‌شوند.

```
error_hist = []
for epoch in range(n_epochs):
    # predictions
    y_hat = logestic_regression(X_train, w)

    # loss
    e = bce(y_train, y_hat)
    error_hist.append(e)

    # gradients
    grads = gradient(X_train, y_train, y_hat)

    # gradient descent
```

```
    w = gradient_decent(w, eta, grads)
    acc = accuracy(y_train, y_hat)
    print(f'Epoch={epoch}, \t E={e}, \t accuracy={acc}')
```

Code 11: (Python)

شکل۳ خروجی تابع Loss را در هر اجرا نمایش می‌دهد.



شکل ۳: خروجی تابع Loss در هر مرحله

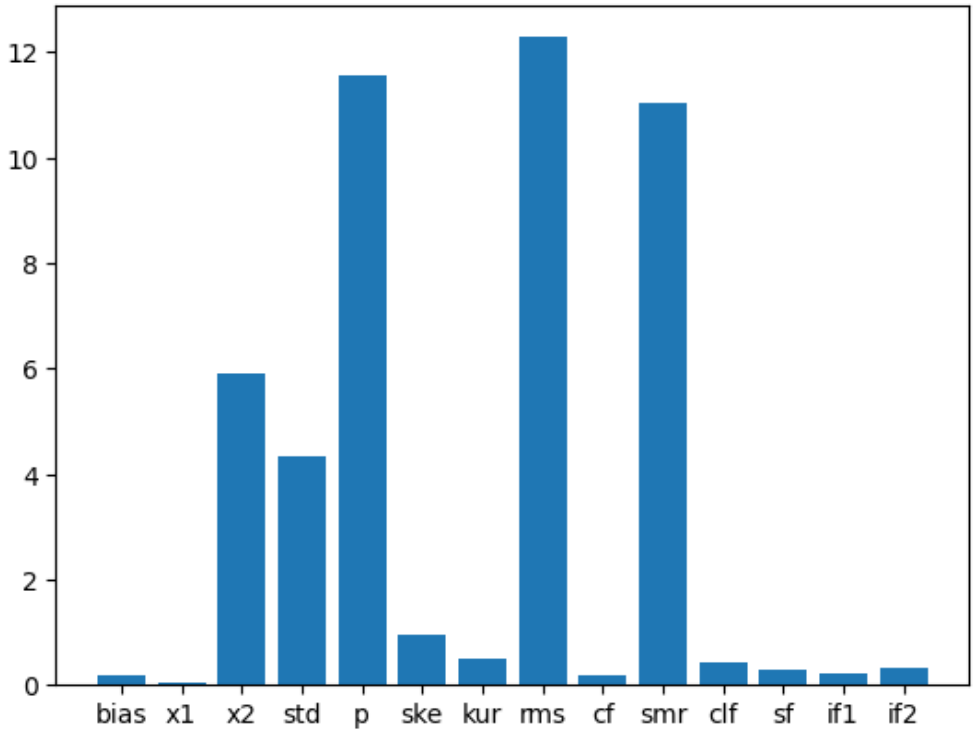
شکل۴ قدر مطلق وزن‌ها را نمایش می‌دهد. دقت خروجی مدل ۸۴ درصد می‌باشد.

۳ سوال سوم

۱.۳ ابتدا هیت مپ ماتریس همبستگی و هیستوگرام پراکندگی ویژگی ها را رسم و تحلیل کنید

همان طور که در Code۱۲ مشاهده می‌شود پس از دانلود دیتاست سطرهایی که حاوی Null هستند حذف می‌شوند و سپس اطلاعات عددی از دیتاست استخراج می‌شود.

```
!gdown 1gWTHhsD52p_0ZjqIE89UHsjfISBltKuC
```

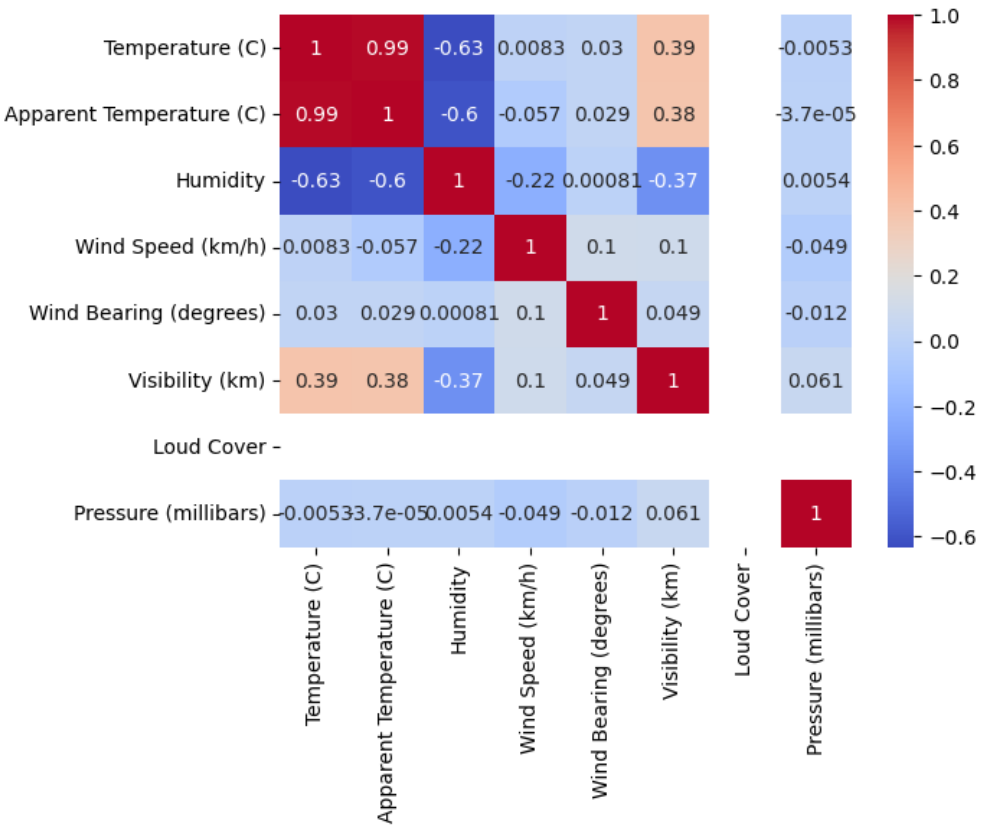


شکل ۴: قدر مطلق وزن ها

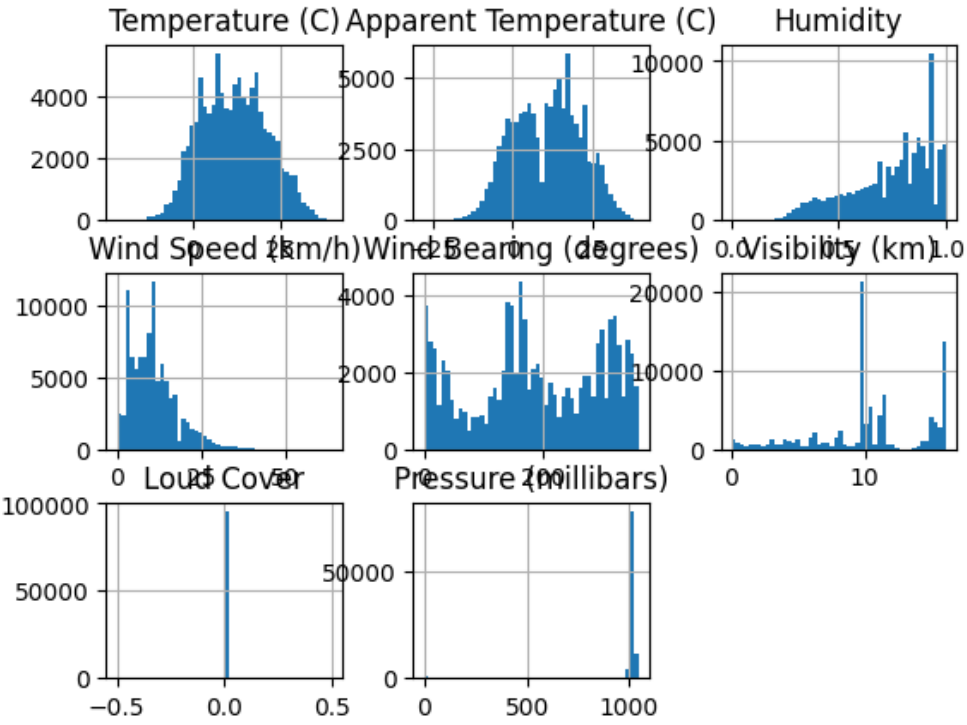
```
weatherHistory = pd.read_csv('weatherHistory.csv')

weatherHistory = weatherHistory.dropna()
numeric_weather_df = weatherHistory.select_dtypes(include=['number'])
print(weatherHistory.keys())
```

Code 12: (Python)

شکل۵ هیت مپ و شکل۶ هیستوگرام را نمایش می‌دهد.


Temperature (C)
Apparent Temperature (C)
Humidity
Wind Speed (km/h)
Wind Bearing (degrees)
Visibility (km)
Loud Cover
Pressure (millibars)
1 0.99 -0.63 0.0083 0.03 0.39 -0.0053
0.99 1 -0.6 -0.057 0.029 0.38 -3.7e-05
-0.63 -0.6 1 -0.22 0.00081 -0.37 0.0054
0.0083 -0.057 -0.22 1 0.1 0.1 -0.049
0.03 0.029 0.00081 0.1 1 0.049 -0.012
0.39 0.38 -0.37 0.1 0.049 1 0.061
-0.0053 -3.7e-05 0.0054 -0.049 -0.012 0.061 1
Temperature (C)
Apparent Temperature (C)
Humidity
Wind Speed (km/h)
Wind Bearing (degrees)
Visibility (km)
Loud Cover
Pressure (millibars)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

شکل ۵: هیت مپ



شکل ۶: هیستوگرام