apl3

vasireddyrevanth2016

February 2022

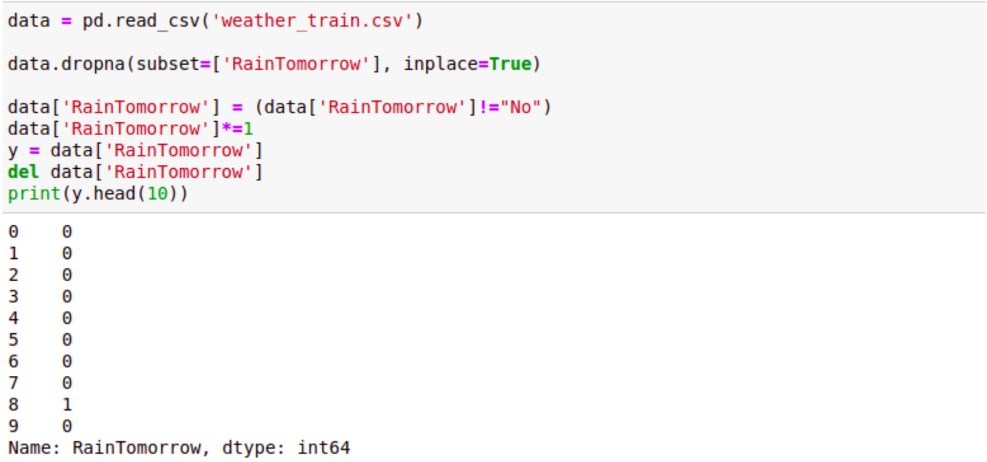
1 Question 1

1.1 Sub Question 1

1.1.1 Code

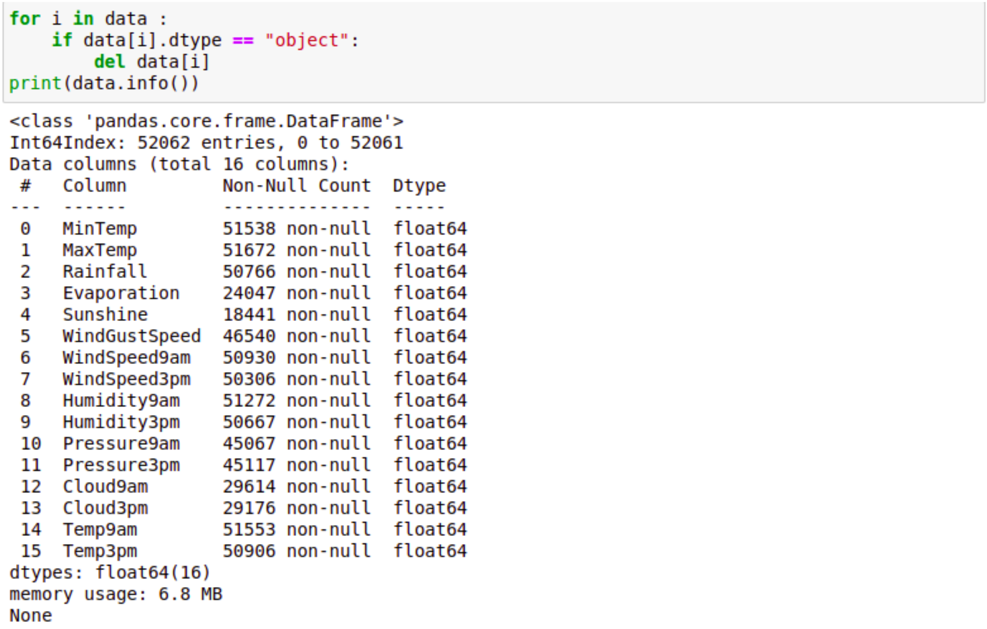
```
data = pd.read_csv('weather_train.csv')

data.dropna(subset=['RainTomorrow'], inplace=True)

data['RainTomorrow'] = (data['RainTomorrow']!="No")
data['RainTomorrow']*=1
y = data['RainTomorrow']
del data['RainTomorrow']
print(y.head(10))

for i in data :
    if data[i].dtype == "object":
        del data[i]
print(data.info())

data.fillna(data.mean(), inplace = True)
print(data.head())

data = (data - data.min())/(data.max() - data.min())
print(data.head())
```

1.1.2 Screenshots



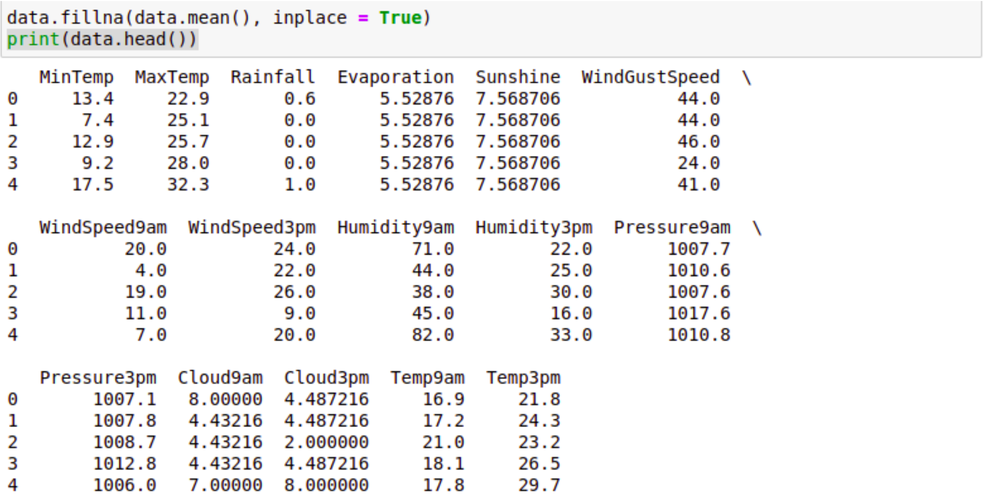
```
data = pd.read_csv('weather_train.csv')

data.dropna(subset=['RainTomorrow'], inplace=True)

data['RainTomorrow'] = (data['RainTomorrow']!="No")
data['RainTomorrow']*=1
y = data['RainTomorrow']
del data['RainTomorrow']
print(y.head(10))
```

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    1
9    0
Name: RainTomorrow, dtype: int64
```



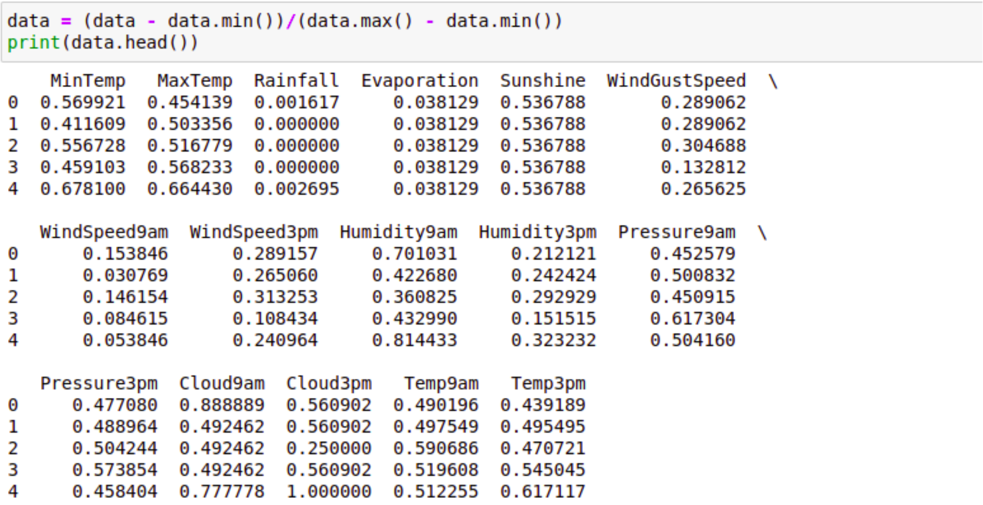
```
for i in data :
    if data[i].dtype == "object":
        del data[i]
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 52062 entries, 0 to 52061
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MinTemp        51538 non-null  float64
 1   MaxTemp        51672 non-null  float64
 2   Rainfall       50766 non-null  float64
 3   Evaporation    24047 non-null  float64
 4   Sunshine       18441 non-null  float64
 5   WindGustSpeed  46540 non-null  float64
 6   WindSpeed9am   50930 non-null  float64
 7   WindSpeed3pm   50306 non-null  float64
 8   Humidity9am    51272 non-null  float64
 9   Humidity3pm    50667 non-null  float64
 10  Pressure9am    45067 non-null  float64
 11  Pressure3pm    45117 non-null  float64
 12  Cloud9am       29614 non-null  float64
 13  Cloud3pm       29176 non-null  float64
 14  Temp9am        51553 non-null  float64
 15  Temp3pm        50906 non-null  float64
dtypes: float64(16)
memory usage: 6.8 MB
None
```



```python
data.fillna(data.mean(), inplace = True)
print(data.head())
```

```
   MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0     13.4     22.9       0.6      5.52876  7.568706           44.0
1      7.4     25.1       0.0      5.52876  7.568706           44.0
2     12.9     25.7       0.0      5.52876  7.568706           46.0
3      9.2     28.0       0.0      5.52876  7.568706           24.0
4     17.5     32.3       1.0      5.52876  7.568706           41.0

   WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  \
0          20.0          24.0         71.0         22.0       1007.7
1           4.0          22.0         44.0         25.0       1010.6
2          19.0          26.0         38.0         30.0       1007.6
3          11.0           9.0         45.0         16.0       1017.6
4           7.0          20.0         82.0         33.0       1010.8

   Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm
0       1007.1   8.00000  4.487216     16.9     21.8
1       1007.8   4.43216  4.487216     17.2     24.3
2       1008.7   4.43216  2.000000     21.0     23.2
3       1012.8   4.43216  4.487216     18.1     26.5
4       1006.0   7.00000  8.000000     17.8     29.7
```



```python
data = (data - data.min())/(data.max() - data.min())
print(data.head())
```

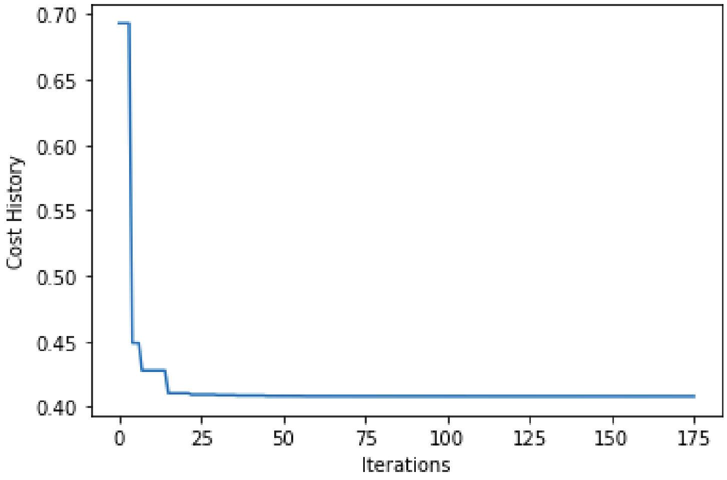
```
    MinTemp   MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0  0.569921  0.454139  0.001617     0.038129  0.536788       0.289062
1  0.411609  0.503356  0.000000     0.038129  0.536788       0.289062
2  0.556728  0.516779  0.000000     0.038129  0.536788       0.304688
3  0.459103  0.568233  0.000000     0.038129  0.536788       0.132812
4  0.678100  0.664430  0.002695     0.038129  0.536788       0.265625

   WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  \
0      0.153846      0.289157     0.701031     0.212121     0.452579
1      0.030769      0.265060     0.422680     0.242424     0.500832
2      0.146154      0.313253     0.360825     0.292929     0.450915
3      0.084615      0.108434     0.432990     0.151515     0.617304
4      0.053846      0.240964     0.814433     0.323232     0.504160

   Pressure3pm  Cloud9am  Cloud3pm   Temp9am   Temp3pm
0     0.477080  0.888889  0.560902  0.490196  0.439189
1     0.488964  0.492462  0.560902  0.497549  0.495495
2     0.504244  0.492462  0.250000  0.590686  0.470721
3     0.573854  0.492462  0.560902  0.519608  0.545045
4     0.458404  0.777778  1.000000  0.512255  0.617117
```

1.2 Sub Question 4

1.2.1 Code

```
data = pd.read_csv('weather_train.csv')

n = lr()
X, y, mean = n.data_clean(data)

m = costing()
w, acc = (m.minCostFun(np.zeros(X.shape[1]+1), X, y, 40000))
# I have included a feature J_his in the class m, to extract J_his.
j = np.array(m.J_his)
print(w)

pyplot.plot(range(j.shape[0]), j)
pyplot.xlabel("Iterations")
pyplot.ylabel("Cost History")
```

1.2.2 Figure



1.2.3 Interpretation

- It took very less number of iterations to reach the below value of the cost. We can also observe that it(optimise.minimize function) is terminating after these few number of the iterations. This kind of looks like a Adaptive Gradient Descent.

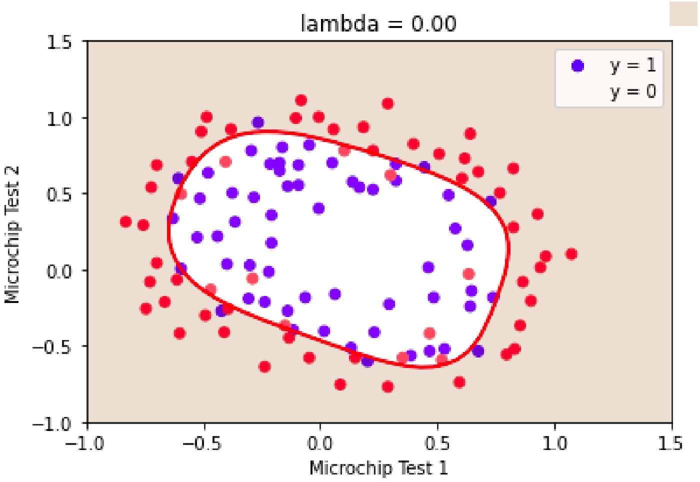
2 Question 2

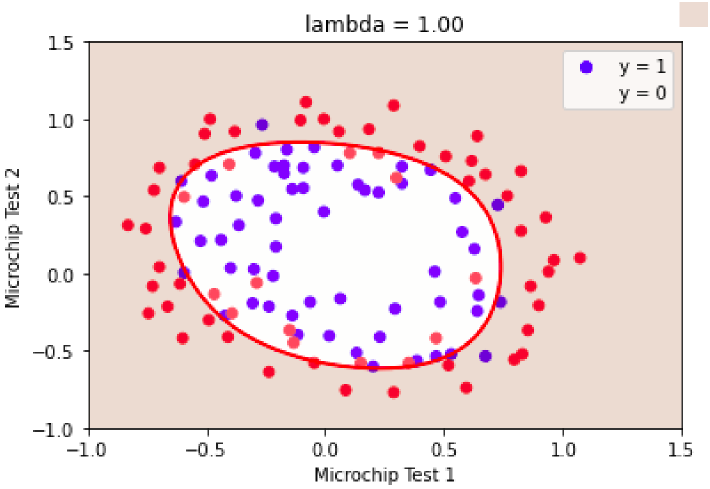
2.1 Code

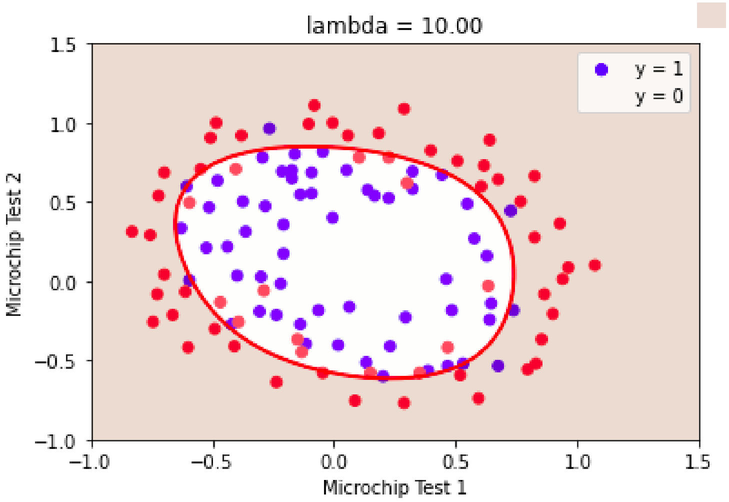
```
data = np.loadtxt('nonlinearClass.txt', delimiter = ',')
X = data[:, :2]
y = data[:, 2]
def f(x, y): #creates degree 4 array
    return np.array([1.0, x, y, x*x, x*y, y*y, x*x*x, x*x*y, x*y*y, y*y*y, x*x*x*x, x*x*x*y,
new_data = []
for i in range(X.shape[0]):
    new_data.append(f(X[i][0], X[i][1]))
X1 = np.transpose(new_data)
X1 = np.transpose(X1[1:])

m = costing()
w, acc = (m.minCostFun(np.zeros(X1.shape[1]+1), X1, y, 40000))

color= []
for i in y:
    if i==1:
        color.append('b')
    else:
        color.append('r')
pyplot.scatter(X[:, 0], X[:, 1], c = color)
x1 = np.linspace(-1, 1, 1000000)

u = np.linspace(-1, 1.5, 50)
v = np.linspace(-1, 1.5, 50)

z = np.zeros((u.size, v.size))
# Evaluate z = w*x over the grid
for i, ui in enumerate(u):
    for j, vj in enumerate(v):
        z[i, j] = np.dot(f(ui, vj), w)

z = z.T

pyplot.contour(u, v, z, levels=[0], linewidths=2, colors='r')
pyplot.contourf(u, v, z, levels=[np.min(z), 0, np.max(z)], cmap='pink', alpha=0.4)

pyplot.xlabel('Microchip Test 1')
pyplot.ylabel('Microchip Test 2')
pyplot.legend(['y = 1', 'y = 0'])
pyplot.grid(False)
pyplot.title('lambda = %0.2f' % 0)
```

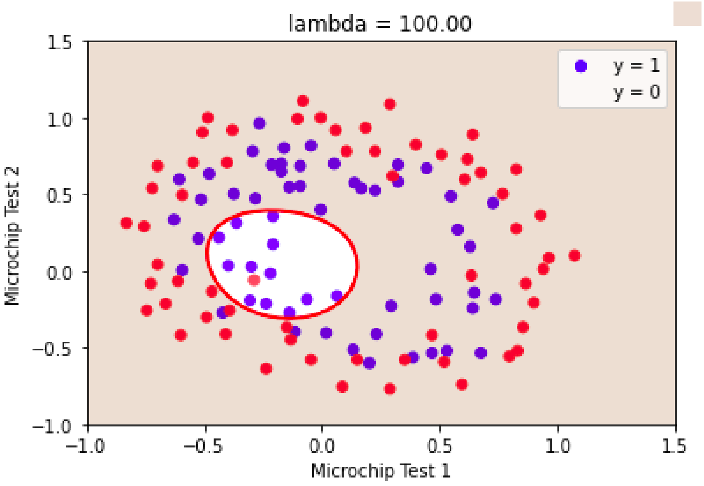
2.2 Figures

```
Text(0.5, 1.0, 'lambda = 0.00')
```









2.3 Interpretation

- We can see that the model is more overfitted for lower values of lambda.
- So, as lambda value increases, the decision boundary deviates from its optimum value for the training data, to a different value; and also size decreases. As, the value of regularization parameter(lambda) increases further, it becomes underfitting and loses main properties derived from the training data.
- Hence , a good value of the parameter is to be selected, so as to not become overfitting or underfitting.