Toprak Verilerine Göre En Uygun Mahsül Ekimi

Oguz Narli

May 2022

1 Giriş

Bu çalışmada logistic regression kullanılarak ortamdaki azot,fosfor,potasyum,sıcaklık,nem,yağmur oranına bakılarak ekilebilecek en uygun mahsulün ne olduğunu makine öğrenimiyle karar veren yöntem geliştirilmiştir. İlgili çalışmada kullanılan veri seti kaggle üzerinden alınmış ve çalışma öncesi veri seti üzerinde bir takım azaltma uygulanmıştır. Tahminler 5 mahsül için yapılmaktadır(pirinç,mısır,nohut,Meksika fasulyesi,güvercin bezelye).

2 Veri Setinin Hazırlanması

İlk olarak gerçekleştirilen pandas kütüphanesi ile csv dosyasından veri seti alınmıştır.

```
import pandas as pd

basepath = "/content/"
crop_data = pd.read_csv(basepath + "Crop_recommendation.csv",usecols=['N','P','K
crop_data.head()
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Figure 1: Sonuç

Veri seti değerleri büyük tam sayılarla olduğundan veriler üzerinde normalizasyon yapılmıması gerekmektedir.Bunun için sayıları sütun içersinde yer alan en büyük tam sayı alınarak,sütun içersindeki her bir değer ile bölünmektedir.

```
import numpy as np
crop_data['N']=crop_data['N']/np.max(crop_data['N'])
crop_data['P']=crop_data['P']/np.max(crop_data['P'])
crop_data['K']=crop_data['K']/np.max(crop_data['K'])
crop_data['temperature']=crop_data['temperature']/np.max(crop_data['temperature'
crop_data['humidity']=crop_data['humidity']/np.max(crop_data['humidity'])
crop_data['ph']=crop_data['ph']/np.max(crop_data['ph'])
crop_data['rainfall']=crop_data['rainfall']/np.max(crop_data['rainfall'])
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	0.90	0.5250	0.505882	0.564654	0.965089	0.733248	0.679714	rice
1	0.85	0.7250	0.482353	0.588742	0.945281	0.793585	0.759162	rice
2	0.60	0.6875	0.517647	0.622113	0.968832	0.884027	0.884124	rice
3	0.74	0.4375	0.470588	0.716403	0.943383	0.787079	0.813451	rice
4	0.78	0.5250	0.494118	0.544383	0.960407	0.860153	0.879948	rice

Figure 2: Sonuç

Bundan sonra yapılan işlemde ilk yapılan işlem string olarak etiketlenmiş değerler bir tam sayı ile id belirlenmesi işlemi yapılmıştır.Devamında ise veri seti eğitilecek ve test veri seti olarak ayrılmıştır.

```
from sklearn.model_selection import train_test_split

crop_data['label'].mask(crop_data['label']=='rice',0,inplace=True)
crop_data['label'].mask(crop_data['label']=='maize',1,inplace=True)
crop_data['label'].mask(crop_data['label']=='chickpea',2,inplace=True)
crop_data['label'].mask(crop_data['label']=='kidneybeans',3,inplace=True)
crop_data['label'].mask(crop_data['label']=='pigeonpeas',4,inplace=True)

train, test = train_test_split(crop_data, test_size=0.2)
train_x,train_y=train.iloc[:,:-1],train.iloc[:,[-1]]
test_x,test_y=test.iloc[:,:-1],test.iloc[:,[-1]]
```

3 Logistic Regression İmplementasyonu

Logistic Regression algoritması girilen parametrelere göre sınıflandırma yapan bir yapar zeka algoritmasıdır.Algoritmanın aşamaları şu şekildedir:

1. İlk olarak veri seti elemanları ağırlık vektörü ile matris çarpımına tabi tutulur ardından bias değeri ile toplanır.

$$\text{Z}= \begin{bmatrix} X_{00} & X_{01} & X_{02} & ... & X_{0N} \\ X_{10} & X_{11} & X_{12} & ... & X_{1N} \\ . & & & & \\ . & & & & \\ . & & & & \\ X_{M0} & X_{M1} & X_{M2} & ... & X_{MN} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_N \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ . \\ . \\ . \\ b_N \end{bmatrix}$$

2. Ağırlık vektörü ile veri setleri çarpıldığından çıkan sonuç sigomid fonksiyonuna sokulur.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

3. Sigmoid fonksiyonuna sokulduktan sonra çıka değer maliyet hesabı yapılır.

$$J(\theta) = -1/m \sum_{i=1}^{m} [y^i \log(h_\theta(x^i)) - (1 - y^i) \log(h_\theta(x^i))]$$

4. Maliyet fonksiyonu bulunduktan sonra gradyan inişi kullanılır ve eğim 0'a yaklaşana kadar devam eder. Bununla beraber ağırlık vektörü elemanları güncellenir.

$$\frac{\partial J(\theta)}{\partial \theta_j} = 1/m \sum_{i=1}^{m} (h_\theta(x) - y^i) x_j^i$$

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Türevin değerin 0 yakalaştıkça bu işlem devam eder.

Yukarki matematiksel işlemlerin python üzerinde uygulanması aşağıda verilmiştir.

1. Sigmoid fonksiyonu:

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

2. Maliyet fonksiyonu:

```
def compute_cost(X, y, theta):
    m = len(y)
    h = sigmoid((X @ theta).to_numpy().astype(float))
    epsilon = 1e-5
    cost = (1/m)*(((-y).T @ np.log(h + epsilon))-((1-y).T @ np.log(1-h + eps
    return cost
```

3. Gradyan inişi:

```
def gradient_descent(X, y, params, learning_rate, iterations):
    m = len(y)
    cost_history = np.zeros((iterations,1))
    for i in range(iterations):
        params = (params.T - ((learning_rate/m) * (X.T @ (sigmoid((X @ param
        cost_history[i] = compute_cost(X, y, params)

    return (cost_history, params)
```

Şimdi bu tanımladığımız fonksiyonları kullanarak veri setimizi eğitelim:

```
gradient_descent(train_x,train_y,parameter,0.05,500)
```

Yukarıdaki kod parçacığından grandyan inişi fonksiyonunu çağırarak veri setimizi eğitmiş oluruz. Fonksiyon parametere olarak eğitilcek veri seti, ağırılık vektörü,öğrenme katsayısı ve eğitim kaç iterasyonda yapılacağı verilmiştir.Tahmin için gerekli girilmesi gereken parametereler ağırlık vektörü ile çarpıp çıkan sonuç yuvarlanarak alınmaktadır.

```
def predict(X, params):
    return np.round(sigmoid(X @ params))
pedict(test_x,parameter)
```