CEMALETTİN HALİL

21040301069

Dersin Hocası:

Dr. Öğretim GÖKALP TULUM

İstanbul Topkapı Üniversitesi

Makine Öğrenmesi

Github Url

https://github.com/jamal-kh/Makine_Ogrenmesi

2024

Diyabet Teşhisi İçin Makine Öğrenimi Algoritmalarının Karşılaştırılması

Diyabet, dünya çapında önemli bir sağlık sorunudur ve doğru teşhis edilmesi, uygun tedavi ve yönetim açısından kritik öneme sahiptir. Makine öğrenimi algoritmaları, diyabet teşhisi için değerli araçlar sağlayabilir. Bu çalışmada, farklı makine öğrenimi sınıflandırma algoritmalarının diyabet teşhisindeki performansını değerlendireceğiz.

Veri Seti ve Ön İşleme: Analizde kullanılacak veri seti, çeşitli sağlık özelliklerini içerir ve diyabet teşhisine ilişkin sonucu gösteren bir hedef değişken içerir. Veri seti eksik değerlerle başa çıkmak için uygun yöntemlerle işlenecek ve özellikler standartlaştırılacaktır.

Kullanılan Algoritmalar: Çalışmada, Naive Bayes, K-En Yakın Komşuluk (K-NN), Çok Katmanlı Algılayıcı (MLP) ve Destek Vektör Makineleri (SVM) gibi yaygın olarak kullanılan sınıflandırma algoritmaları incelenecektir.

Amaç ve Beklentiler: Bu çalışmanın amacı, diyabet teşhisi için kullanılan farklı sınıflandırma algoritmalarının performansını karşılaştırmak ve en etkili olanını belirlemektir. Doğru teşhis algoritmalarının belirlenmesi, sağlık profesyonellerine ve hastalara daha iyi bir tedavi ve yönetim stratejisi sunabilir.

Yöntemler ve Analiz: Veri seti üzerinde çeşitli makine öğrenimi algoritmaları eğitilecek ve test edilecektir. Her bir algoritmanın performansı, hassasiyet, özgünlük, doğruluk, F1-skoru ve ROC eğrisi altında alan (AUC) gibi metrikler kullanılarak değerlendirilecektir.

Naive Bayes Sınıflandırıcısı:

```
# Naive Bayes modeli eğit ve tahmin yap
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)

```

Bu kısımda, GaussianNB sınıfından bir Naive Bayes modeli oluşturulur. Model, eğitim verileri (X_train ve y_train) kullanılarak eğitilir. Daha sonra, eğitilmiş model kullanılarak test verileri (X_test) üzerinde tahmin yapılır ve y_pred_nb değişkeninde tahmin edilen sınıflar elde edilir.

Sonuç:

```
Naive Bayes Sınıflandırıcısı
[[120  30]
 [ 33  48]]
              precision    recall  f1-score   support

           0       0.78      0.80      0.79       150
           1       0.62      0.59      0.60        81

    accuracy                           0.73       231
   macro avg       0.70      0.70      0.70       231
weighted avg       0.73      0.73      0.73       231
```

K-En Yakın Komşuluk (K-NN) Sınıflandırıcısı:

```python
# K-NN modeli ve GridSearchCV ile en iyi k değerini bulma
param_grid = {'n_neighbors': np.arange(1, 50)}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, param_grid, cv=5)
knn_cv.fit(X_train, y_train)

# En iyi k değeri ile model eğitimi ve tahmin yapma
best_k = knn_cv.best_params_['n_neighbors']
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

```

Bu kısımda, KNeighborsClassifier sınıfından bir K-NN modeli oluşturulur. GridSearchCV yöntemi kullanılarak en iyi k değeri (best_k) bulunur. Daha sonra, en iyi k değeri ile tekrar bir K-NN modeli oluşturulur, eğitilir ve test verileri üzerinde tahmin yapılır.

Sonuç:

```
K-En Yakın Komşuluk Sınıflandırıcısı (En iyi k=12)
[[130  20]
 [ 43  38]]
              precision    recall  f1-score   support

           0       0.75      0.87      0.80       150
           1       0.66      0.47      0.55        81

    accuracy                           0.73       231
   macro avg       0.70      0.67      0.68       231
weighted avg       0.72      0.73      0.71       231
```

Çok Katmanlı Algılayıcı (MLP) Sınıflandırıcısı:

```python
# MLP modeli eğitimi ve tahmin yapma
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
mlp.fit(X_train, y_train)
y_pred_mlp = mlp.predict(X_test)

```

Bu kısımda, MLPClassifier sınıfından bir MLP modeli oluşturulur. Model, belirli parametrelerle (örneğin, gizli katman boyutu, maksimum iterasyon sayısı) ayarlanır, eğitilir ve test verileri üzerinde tahmin yapılır.

sonuç:

```
Çok Katmanlı Algılayıcı (MLP) Sınıflandırıcısı
[[127  23]
 [ 35  46]]
              precision    recall  f1-score   support

           0       0.78      0.85      0.81       150
           1       0.67      0.57      0.61        81

    accuracy                           0.75       231
   macro avg       0.73      0.71      0.71       231
weighted avg       0.74      0.75      0.74       231
```

Destek Vektör Makineleri (SVM) Sınıflandırıcısı:

```python
# MLP modeli eğitimi ve tahmin yapma
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
mlp.fit(X_train, y_train)
y_pred_mlp = mlp.predict(X_test)

```

Bu kısımda, SVC sınıfından bir SVM modeli oluşturulur. Model, eğitilir ve test verileri üzerinde tahmin yapılır.

Sonuç:

```
Destek Vektör Makineleri (SVM) Sınıflandırıcısı
[[130  20]
 [ 41  40]]
              precision    recall  f1-score   support

           0       0.76      0.87      0.81       150
           1       0.67      0.49      0.57        81

    accuracy                           0.74       231
   macro avg       0.71      0.68      0.69       231
weighted avg       0.73      0.74      0.72       231
```

ROC EĞRİSİ: