Veri Yoğun Uygulamalar (Spark) Ödev Raporu

Tugay Talha İçen

Mart, 2024

1 Giriş

Bu rapor, Kaliforniya Konut Fiyatları veri kümesi kullanılarak ev fiyatlarını tahmin etmek için PySpark kullanılarak bir regresyon modeli oluşturma sürecini ve sonuçlarını sunmaktadır. Görevin amacı ve kapsamı, veriyi analiz etmek, özellikleri seçmek, makine öğrenimi için veriyi hazırlamak, bir makine öğrenimi modeli oluşturmak ve performansını ölçmektir.

2 Veri Seti

Bu görev için Kaliforniya Konut Fiyatları veri kümesi kullanılmıştır. Bu veri kümesi, boylam, enlem, konut ortanca yaşı, toplam oda sayısı, toplam yatak odası sayısı, nüfus, hanehalkı, ortanca gelir ve ortanca ev değeri gibi çeşitli özellikler içermektedir.

3 Kod Genel Bakışı

code.ipynb defteri, PySpark kullanarak görevin görevlerinin uygulanmasını içerir. Ana bölümleri şunlardır:

1. PySpark ortamını kurma ve gerekli kütüphaneleri içe aktarma.
2. Veri kümesini yükleme ve yapısını ve içeriğini analiz etme.
3. Veriyi ön işleme, eksik değerleri işleme ve kategorik değişkenleri kodlama.
4. Özellik seçimi için Recursive Feature Elimination (RFE).
5. Doğrusal Regresyon, Rastgele Orman, Gradyan Artırılmış Ağaç, Karar Ağacı, Genelleştirilmiş Doğrusal Regresyon ve Faktörizasyon Makineleri gibi çeşitli algoritmaları kullanarak regresyon modelleri oluşturma.
6. Ortalama Kare Hatası (MSE), Kök Ortalama Kare Hatası (RMSE) ve R-kare gibi ölçütler kullanarak model performansını değerlendirme.

7. En iyi performans gösteren modeli seçme ve tüm veri kümesi üzerinde eğitme.

8. Eğitilmiş modeli gelecekteki kullanım için kaydetme.

4 Kod Parçaları

Aşağıda, uygulamanın ana kod parçalarından bazıları bulunmaktadır:

4.1 Kütüphanelerin İçe Aktarılması ve PySpark Ortamının Kurulması

```
!pip install pyspark
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, StandardScaler, OneHotEncoder, S
from pyspark.ml.regression import LinearRegression, RandomForestRegressor, GBTR
from pyspark.ml.evaluation import RegressionEvaluator
import numpy as np
```

4.2 Verinin Yüklenmesi ve Analiz Edilmesi

Bu bölümde, bir SparkSession başlatılır, veri kümesi bir CSV dosyasından okunur, şeması ve ilk 5 satırı görüntülenir. Ardından, veri kümesindeki toplam satır sayısı hesaplanır ve "ocean_proximity" sütunundaki benzersiz değerler yazdırılır. Son olarak, veri kümesindeki her sütundaki eksik değerlerin oranı hesaplanır.

```
# SparkSession olu tur
spark = SparkSession.builder.master('local').appName("HousePricePrediction").get

# CSV dosyas ndan veriyi y kle
data = spark.read.csv("housing.csv", header=True, inferSchema=True)

# Veriyi analiz et
data.printSchema()
data.show(5)
print("Toplam sat r say s :", data.count())

for col in ["ocean_proximity"]:
    print(f"{col}'s tunundaki benzersiz de erler:")
    data.select(col).distinct().show()

for col in data.columns:
    print(f"{col}'s tunundaki eksik de er oran :{data.filter(data[col].isNul
```

4.3 Veri Ön İşleme

Bu parça, veri ön işleme görevlerini gerçekleştirir; kategorik değişkenleri tek-etiketleme kodu ile işler, "total_bedrooms" sütunundaki eksik değerleri ele alır ve özellikleri StandardScaler kullanarak ölçeklendirir.

```
# Kategorik s tunlar i in tek-etiketleme kodu yap
indexer = StringIndexer(inputCol="ocean_proximity", outputCol="ocean_proximity_in
data = indexer.fit(data).transform(data)
encoder = OneHotEncoder(inputCol="ocean_proximity_index", outputCol="ocean_proxi
encModel = encoder.fit(data)
data = encModel.transform(data)


data.show(5)
print("Toplam sat r say s :", data.count())

# 'total_bedrooms' s tunundaki eksik de erleri ele al
data = data.fillna(data.approxQuantile("total_bedrooms", [0.5], 0.001)[0], subse
```

4.4 Özellik Seçimi ve Veri Hazırlığı

Bu bölüm, Recursive Feature Elimination (RFE) kullanarak en önemli özellikleri seçmek için bir rfe_feature_selection işlevini tanımlar ve ardından RFE'yi en üst 8 özellik seçmek için uygular ve özellikleri bir vektörde birleştirir ve ölçeklendirir.

```
def rfe_feature_selection(data, features, target_col, num_features, estimator):
    # Kalan    zellikler    ve se ilen    zellikler    ba lat l r
    remaining_features = features
    selected_features = []

    while len(selected_features) < num_features:
        # Kalan    zellikler    i in VectorAssembler olu tur
        assembler = VectorAssembler(inputCols=remaining_features, outputCol="fea
        data_assembled = assembler.transform(data)

        # Random Forest modelini e it
        rf = estimator(labelCol=target_col, featuresCol="features")
        model = rf.fit(data_assembled)

        #    zellik    nemlerini    al
        importances = model.featureImportances

        # SparseVector'    yo un temsile d n    t r
        importances_dense = importances.toArray()
```

```
            # En az nemli zellii bul
            least_important_feature_index = np.argmin(importances_dense)

            least_important_feature = remaining_features[least_important_feature_inde


            # Kalan zelliklerden en az nemli zellii kar
            remaining_features.remove(least_important_feature)

            # En az nemli zellii se ilen zelliklere ekle
            selected_features.append(least_important_feature)

    # Son zellikleri se
    return data.select(selected_features + [target_col])

# 'ocean_proximity_encoded' ve 'ocean_proximity' haricindeki zellikleri se
features = [col for col in data.columns if col != "ocean_proximity_encoded" and

# Gereksiz zellikleri ele
num_features = 8 # K k De erler ok k t sonu lar veriyor
data_filtered = rfe_feature_selection(data, features, "median_house_value", num

print(data_filtered.columns)

# 'median_house_value' haricindeki zellikleri se
features = [col for col in data_filtered.columns if col != "median_house_value"]
if "ocean_proximity_index" in features:
    features.remove("ocean_proximity_index")
    features.append("ocean_proximity_encoded")

# zellikleri tek bir vekt rde birle tirmek i in bir VectorAssembler olu tu
assembler = VectorAssembler(inputCols=features, outputCol="features")
data = assembler.transform(data)

# zellikleri leklendir
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures")
data = scaler.fit(data).transform(data)

# Veriyi e itim ve test setlerine b le
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
```

4.5 Regresyon Modelleri Oluşturma, Eğitme ve Değerlendirme

Bu bölümde, regresyon modelleri ve karşılık gelen PySpark modellerinin bir

listesi tanımlanır. Döngü, her bir model üzerinde döner, onu eğitir, test verisi üzerinde tahminler yapar ve Ortalama Kare Hata (MSE), Kök Ortalama Kare Hata (RMSE) ve R-kare ölçütlerini kullanarak performansını değerlendirir.

```
# Regresyon modelleri olu tur
models = [
    ("Do rusal Regresyon", LinearRegression(labelCol="median_house_value", feat
    ("Rastgele Orman", RandomForestRegressor(labelCol="median_house_value", feat
    ("Gradyan Art r lm  A a ", GBTRegressor(labelCol="median_house_value"
    ("Karar A ac ", DecisionTreeRegressor(labelCol="median_house_value", featu
    ("Genelle tirilmi Do rusal Regresyon", GeneralizedLinearRegression(label
    ("Fakt rizasyon Makineleri", FMRegressor(labelCol="median_house_value", fea
]
evaluator = RegressionEvaluator(labelCol="median_house_value", predictionCol="pr

for name, model in models:
    model = model.fit(train_data)
    predictions = model.transform(test_data)
    mse = evaluator.evaluate(predictions, {evaluator.metricName: "mse"})
    rmse = evaluator.evaluate(predictions, {evaluator.metricName: "rmse"})
    r2 = evaluator.evaluate(predictions, {evaluator.metricName: "r2"})
    print(f"Model: {name}")
    print(f"    MSE: {mse}, RMSE: {rmse}, R-kare: {r2}")
```

4.6 En İyi Model Performansının Değerlendirilmesi

Bu parça, en iyi performans gösteren modeli (Gradyan Artırılmış Ağaç) tüm veri kümesi üzerinde eğitir, MSE, RMSE, MAE, R-kare ve açıklanan varyans gibi çeşitli metrikler kullanarak performansını değerlendirir ve sonuçları yazdırır.

```
# En iyi modeli e it
best_model = GBTRegressor(labelCol="median_house_value", featuresCol="scaledFeat
best_model = best_model.fit(train_data)

# En iyi modeli de erlendir
evaluator = RegressionEvaluator(labelCol="median_house_value", predictionCol="pr
predictions = best_model.transform(test_data)
mse = evaluator.evaluate(predictions, {evaluator.metricName: "mse"})
rmse = evaluator.evaluate(predictions, {evaluator.metricName: "rmse"})
mae = evaluator.evaluate(predictions, {evaluator.metricName: "mae"})
r2 = evaluator.evaluate(predictions, {evaluator.metricName: "r2"})
explained_variance = evaluator.evaluate(predictions, {evaluator.metricName: "var


# Sonu lar  yazd r
print(f"En  yi  Model: Gradyan Art r lm  A a ")
```

```
print(f"----MSE: {mse}, RMSE: {rmse}, MAE: {mae}, R-kare: {r2}, A    klanan Vary
```

4.7 En İyi Modelin Kaydedilmesi

Son olarak, en iyi model (GBTRegressor) tüm veri kümesi üzerinde eğitilir ve gelecekteki kullanım için kaydedilir. Model, "best_model" adlı bir dizine kaydedilir ve daha kolay dağıtım için sıkıştırılır. Kaynakları serbest bırakmak için SparkSession durdurulur.

```
# En iyi modeli t m veri k mesi    zerinde    e it
best_model = GBTRegressor(labelCol="median_house_value", featuresCol="scaledFeat
best_model = best_model.fit(data)

# Modeli kaydet
best_model.save("best_model")

# Modeli s k    t r
import shutil
shutil.make_archive("best_model", 'zip', "best_model")

# SparkSession durdur
spark.stop()
```

5 Sonuçlar

Farklı regresyon modellerinin performansı test verisi kullanılarak değerlendirildi. Gradyan Artırılmış Ağaç modeli, aşağıdaki metriklerle en iyi sonuçları elde etti:

- MSE: 3.210.917.002,30
- RMSE: 56.664,95
- MAE: 39.837,33
- R-kare: 0,768
- Açıklanan Varyans: 10.334.394.537,06

6 Sonuç

Sonuç olarak, Kaliforniya'da ev fiyatlarını tahmin etmek için PySpark kullanılarak bir regresyon modeli başarıyla oluşturuldu. Gradyan Artırılmış Ağaç algoritması, değerlendirilen modeller arasında en iyi performansı gösterdi. Eğitilmiş model, Kaliforniya'daki ev fiyatlarını tahmin etmek için daha fazla kullanılabilir.