Dokumentacja Skryptu Analizy Regresji Danych Spotify

Michał Herka, Daniel Brzezicki

Wprowadzenie

Ten skrypt przeprowadza analizę regresji na danych Spotify, wykorzystując bibliotekę scikit-learn. Celem jest przewidywanie cechy 'streams' - ilości odtworzeń na podstawie wybranych cech wejściowych.

Zależności

- pandas
- numpy
- scikit-learn (sklearn)
- Matplotlib

Enumy

RegressionMethod

- `SINGLE`: Analiza pojedynczej regresji
- `TRAIN_TEST_SPLIT`: Analiza regresji z podziałem na zestaw treningowy i testowy
- `KFOLD`: Analiza regresji z użyciem K-krotnego podziału krzyżowego

RegressorType

- `DECISION_TREE`: Drzewo decyzyjne
- `RANDOM_FOREST`: Las losowy

Funkcje

`singleRegressorMAE(regressor, x_train, y_test)`

Dopasowuje dostarczony regresor do danych treningowych (`x_train`, `y_test`) i zwraca średni błąd bezwzględny (MAE) na danych testowych.

`trainTestSplitRegressorMAE(regressor, df_features, df_main_feature, test_size=0.5)`

Dzieli dane na zestaw treningowy i testowy, dopasowuje regresor i zwraca MAE na zestawie testowym.

`kfoldRegressorMAE(regressor, df_features, df_main_feature, n_splits=5)`

Przeprowadza K-krotny podział krzyżowy, dopasowuje regresor i zwraca maksymalne MAE we wszystkich foldach.

`getMaeValues(dataMethod: RegressionMethod, regressorType: RegressorType, X, y, depth=5)`

Generuje wartości MAE dla różnych głębokości regresora na podstawie określonej metody regresji i typu regresora.

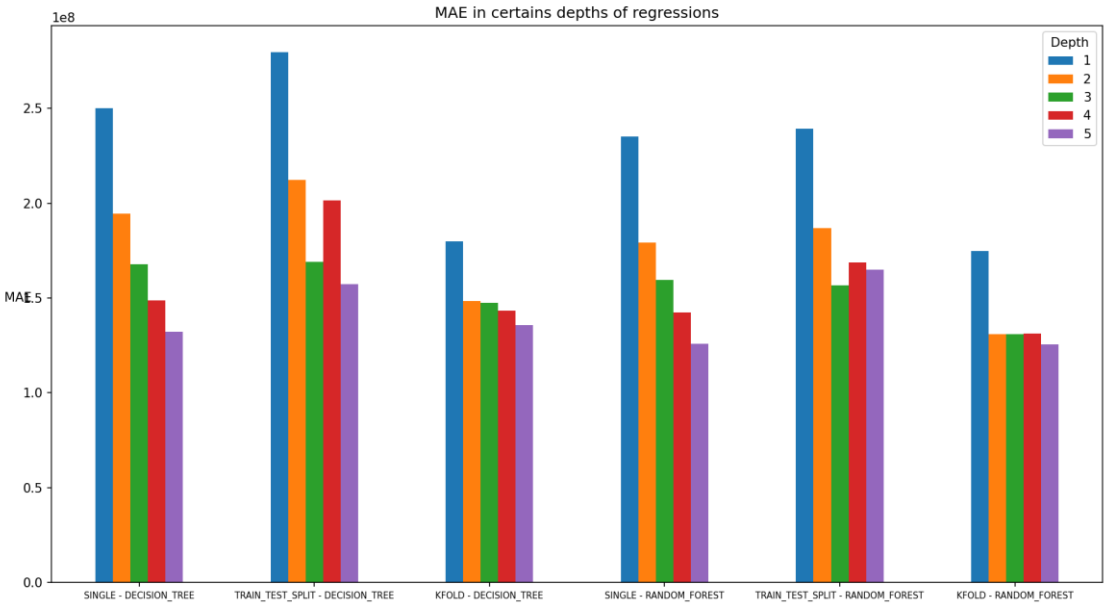
`predictValueByFeatures(dataFrame, features, mainFeature)`

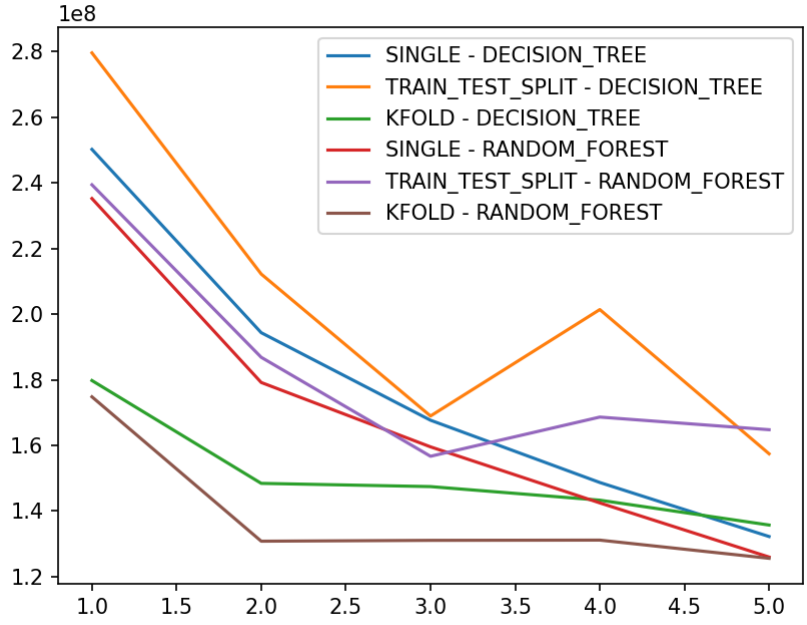
Przetwarza dane, przeprowadza analizę regresji dla różnych metod i typów, a następnie przedstawia wyniki za pomocą wykresów słupkowych i liniowych.

Przykład Użycia

```
result = predictValueByFeatures(pd.read_csv('data/spotify-2023.csv', encoding='latin-1'), ['artist_count','released_year','in_apple_playlists','in_spotify_playlists','in_spotify_charts','danceability_%', 'streams'], 'streams')
```

Wyjście z dobraniem powyższych cech





Decision Tree

W przypadku pojedynczego drzewa decyzyjnego, obserwuje się tendencję do poprawy dokładności modelu wraz z zwiększaniem głębokości drzewa. Warto jednak zauważyć, że głębokość drzewa równa 5 wydaje się być optymalnym wyborem, gdyż dalsze zwiększanie głębokości nie przynosi już tak znaczącej poprawy. Wyniki uzyskane na zbiorze testowym (TRAIN_TEST_SPLIT) różnią się od wyników na zbiorze treningowym, co może sugerować pewne przetrenowanie modelu. Optymalna

głębokość drzewa dla tego zbioru wydaje się być mniejsza niż dla pojedynczego drzewa.

Random Forest

Random Forest, będący ensemblem drzew decyzyjnych, wykazuje ogólnie lepszą zdolność do generalizacji niż pojedyncze drzewo. Otrzymane wyniki są bardziej stabilne i mniej podatne na przetrenowanie. W przypadku Random Forest również zauważa się tendencję do poprawy dokładności wraz z zwiększaniem głębokości drzewa, ale różnice między kolejnymi głębokościami są mniejsze niż dla pojedynczego drzewa.

Porównanie między pojedynczym drzewem a Random Forest:

Random Forest uzyskuje niższe wartości błędów średnich bezwzględnych (MAE) niż pojedyncze drzewo decyzyjne dla wszystkich głębokości drzewa i dla wszystkich rodzajów podziałów danych (SINGLE, TRAIN_TEST_SPLIT, KFOLD). Wyniki na zbiorze KFOLD są najbardziej stabilne, co potwierdza, że Random Forest jest bardziej odporny na różnice między podziałami zbioru danych.

Wnioski ogólne:

Random Forest wydaje się być bardziej wszechstronnym modelem do tego zadania, oferując lepszą zdolność do generalizacji niż pojedyncze drzewo. Optymalna głębokość drzewa może zależeć od specyfiki danych i zadania. Warto przeprowadzić dalsze eksperymenty, aby zoptymalizować parametry modelu. Przy analizie wyników ważne jest również zwrócenie uwagi na ewentualne przetrenowanie modelu, co można zaobserwować na zbiorze testowym w przypadku pojedynczego drzewa decyzyjnego. Regularyzacja modelu lub ograniczenie głębokości drzewa może pomóc w tym przypadku.