Predicting Product Sales

thêm thư viện

- `pandas` (pd): Thư viện để xử lý và phân tích dữ liệu. Dùng để thao tác với các cấu trúc dữ liệu như DataFrame.
- `numpy` (np): Thư viện dùng cho các phép toán mảng, tính toán khoa học.
- `scikit-learn (sklearn)`: Thư viện phổ biến cho học máy:

- `train_test_split`: Chia dữ liệu thành bộ dữ liệu huấn luyện và kiểm tra.
- `cross_val_score`: Đánh giá mô hình bằng phương pháp kiểm tra chéo.
- `LabelEncoder`: Biến đổi các dữ liệu phân loại thành các giá trị số.
- `StandardScaler`: Tiến hành chuẩn hóa (standardization) các đặc trưng của dữ liệu.
- `RandomForestRegressor`: Mô hình học máy Random Forest cho bài toán hồi quy. `mean_squared_error`, `mean_absolute_error`, `r2_score`: Các chỉ số đánh giá hiệu quả mô hình.
- `xgboost` (xgb): Một thư viện học máy mạnh mẽ cho các bài toán dự đoán, đặc biệt hiệu quả với dữ liệu lớn.
- `statsmodels` (ARIMA): Thư viện dành cho phân tích chuỗi thời gian, trong đó ARIMA là một mô hình phổ biến để dự báo chuỗi thời gian.
- `prophet`: Một công cụ dự báo chuỗi thời gian được Facebook phát triển, thường dùng cho dữ liệu có tính mùa vụ hoặc xu hướng.
- `surprise`: Thư viện xây dựng các hệ thống đề xuất, sử dụng để dự đoán các sở thích của người dùng. -`Dataset`: Chứa dữ liệu cho các mô hình học máy trong hệ thống đề xuất. - `Reader`: Đọc dữ liệu và chuyển thành định dạng mà thư viện -`Surprise` có thể sử dụng. -`SVD`: Một phương pháp học máy dùng cho hệ thống đề xuất dựa trên phân tích giá trị kỳ vọng (Singular Value Decomposition).
- `joblib`: Thư viện lưu và tải mô hình đã được huấn luyện, giúp việc triển khai dễ dàng hơn.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
import xgboost as xgb
from statsmodels.tsa.arima.model import ARIMA
from prophet import Prophet
import joblib
from surprise import Dataset, Reader, SVD
from surprise.model_selection import train_test_split as
surprise_train_test_split
```

đọc file .csv

- Đọc dữ liệu từ tệp CSV vào DataFrame của pandas. Lưu ý thay đổi path file để chạy

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

step 1: chuyển bị data

Xử lý các giá trị bị thiếu và giá trị ngoại lai

```
data.fillna(data.mean(numeric_only=True), inplace=True)
data = data[data['sales'] > 0]
```

- `fillna`: Thay thế các giá trị thiếu (NaN) bằng giá trị trung bình của các
- `cột số`.´· Loại bỏ ngoại lai: Bỏ các dòng dữ liệu có giá trị sales <= 0, vì các giá trị này có thể là dữ liệu sai hoặc không hợp lệ. Chuyển đổi đặc trưng phân loại thành giá trị số

Chuyển đổi các tính năng phân loại thành các giá trị số

```
le = LabelEncoder()
for col in data.select_dtypes(include=['object']).columns:
    data[col] = le.fit_transform(data[col])
```

-`LabelEncoder`: Chuyển các cột phân loại (chẳng hạn như tên sản phẩm, loại sản phẩm) thành các giá trị số để mô hình học máy có thể xử lý

step 2:Kỹ thuật tính năng

```
data['product_popularity'] = data['number_of_reviews'] *
data['average_rating']
data['customer_lifetime_value'] = data['purchase_frequency'] *
data['price']
data['seasonality_factor'] = pd.to_datetime(data['date']).dt.month
```

Tạo các đặc trưng mới:

- `product_popularity`: Sự phổ biến của sản phẩm, tính bằng cách nhân số lượng đánh giá và điểm đánh giá trung bình.

- `customer_lifetime_value`: Giá trị khách hàng, tính bằng tần suất mua hàng nhân với giá sản phẩm.
- `seasonality_factor`: Yếu tố mùa vụ, được tạo ra từ tháng trong năm, giả định rằng mùa vụ có ảnh hưởng đến doanh số.

step 3: chia rẽ dữ liệu

```
X = data.drop(['sales', 'date'], axis=1)
y = data['sales']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Chia dữ liệu thành các đặc trưng (X) và biến mục tiêu (sales).

- `train_test_split`: Chia dữ liệu thành tập huấn luyện và tập kiểm tra (80% huấn luyện, 20% kiểm tra).

Chuẩn hóa các tính năng

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- `StandardScaler`: Chuẩn hóa các đặc trưng sao cho giá trị có trung bình bằng 0 và độ lệch chuẩn bằng 1, giúp mô hình học máy hoạt động hiệu quả hơn.

step 4 : mô hình

Random Forest

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

- Xây dựng mô hình Random Forest cho bài toán hồi quy, với 100
- cây quyết định.

`fit`: Huấn luyện mô hình với tập huấn luyện.

`predict`: Dự đoán giá trị doanh số cho tập kiểm tra.

Evaluation for Random Forest

```
rf_mse = mean_squared_error(y_test, rf_predictions)
rf_mae = mean_absolute_error(y_test, rf_predictions)
print("Random Forest - MSE:", rf_mse, "MAE:", rf_mae)
```

`mean_squared_error (MSE)` và `mean_absolute_error (MAE)`: Đo lường độ chính xác của mô hình bằng cách tính toán sai số giữa giá trị thực tế và giá trị dự đoán.

XGBoost

```
xgb_model = xgb.XGBRegressor(objective='reg:squarederror',
n_estimators=100)
xgb_model.fit(X_train, y_train)
xgb_predictions = xgb_model.predict(X_test)
```

- Tạo mô hình `XGBoost` cho bài toán hồi quy.
- `objective='reg:squarederror'`: Chỉ định rằng bài toán là một bài toán hồi quy với sai số bình phương.

Evaluation for XGBoost

```
xgb_mse = mean_squared_error(y_test, xgb_predictions)
xgb_mae = mean_absolute_error(y_test, xgb_predictions)
print("XGBoost - MSE:", xgb_mse, "MAE:", xgb_mae)
```

- Tương tự như Random Forest, đánh giá XGBoost bằng MSE và MAE.

```
# Save the best model
joblib.dump(xgb_model, 'xgb_model.pkl')
```

- `joblib.dump`: Lưu mô hình XGBoost vào tệp .pkl để sử dụng lại sau này

Step 5: Phân tích chuỗi thời gian

```
data['date'] = pd.to_datetime(data['date'])
data.set_index('date', inplace=True)
```

ARIMA

```
arima_model = ARIMA(data['sales'], order=(1, 1, 1))
arima_results = arima_model.fit()
forecast_arima = arima_results.forecast(steps=12)
print("ARIMA Forecast:\n", forecast_arima)
```

- `ARIMA`: Mô hình phân tích chuỗi thời gian, ở đây sử dụng thông số (1, 1, 1) cho phần AR, I và MA. Mô hình này sẽ dự đoán 12 bước tiếp theo (12 tháng).

Prophet

```
prophet_data = data.reset_index().rename(columns={'date': 'ds',
'sales': 'y'})
```

```
model_prophet = Prophet()
model_prophet.fit(prophet_data)
future_dates = model_prophet.make_future_dataframe(periods=12,
freq='M')
forecast_prophet = model_prophet.predict(future_dates)
print("Prophet Forecast:\n", forecast_prophet[['ds',
'yhat']].tail(12))
```

- `Prophet`: Sử dụng Prophet để dự báo doanh số trong tương lai. Chuyển đổi dữ liệu để phù hợp với yêu cầu của Prophet (cột date thành ds và sales thành y).

Step 6: Hệ thống đề xuất

```
reader = Reader(rating_scale=(1, 5))
df_surprise = pd.DataFrame({
    'user_id': data['user_id'],
    'product_id': data['product_id'],
    'rating': data['average_rating']
})
data_surprise = Dataset.load_from_df(df_surprise[['user_id',
'product_id', 'rating']], reader)
trainset, testset = surprise_train_test_split(data_surprise,
test_size=0.2)
svd_model = SVD()
svd_model.fit(trainset)
predictions = svd_model.test(testset)
```

- `Reader và Dataset`: Chuyển đổi dữ liệu thành định dạng mà thư viện Surprise có thể sử dụng cho hệ thống đề xuất.
- `surprise_train_test_split`: Chia dữ liệu thành bộ huấn luyện và kiểm tra cho hệ thống đề xuất.

```
svd_model = SVD()
svd_model.fit(trainset)
predictions = svd_model.test(testset)
```

- `SVD`: Phương pháp phân tích ma trận để tạo ra một mô hình gợi ý cho người dùng

```
from surprise.accuracy import rmse
rmse(predictions)
```

- `rmse`: Tính toán lỗi căn bậc hai trung bình (Root Mean Squared Error) để đánh giá hiệu quả của hệ thống đề xuất.

```
# Deployment placeholder
print("Models are ready for deployment.")
```

in kết quả

source code github: https://github.com/HUyEsona/ML-project_-Predicting-Product-Sales.git