Adi-Soyadi : Mohammad Zaaroura

Ogrenci No: 202013171126

4. sinif 1.ogretim(ders icin 2.ogretim ile)

yüksek düzey programlama

Proje : Predict Future Sales

Doç. Dr. Hasan Temurtaş

Project

-Predict Future Sales-

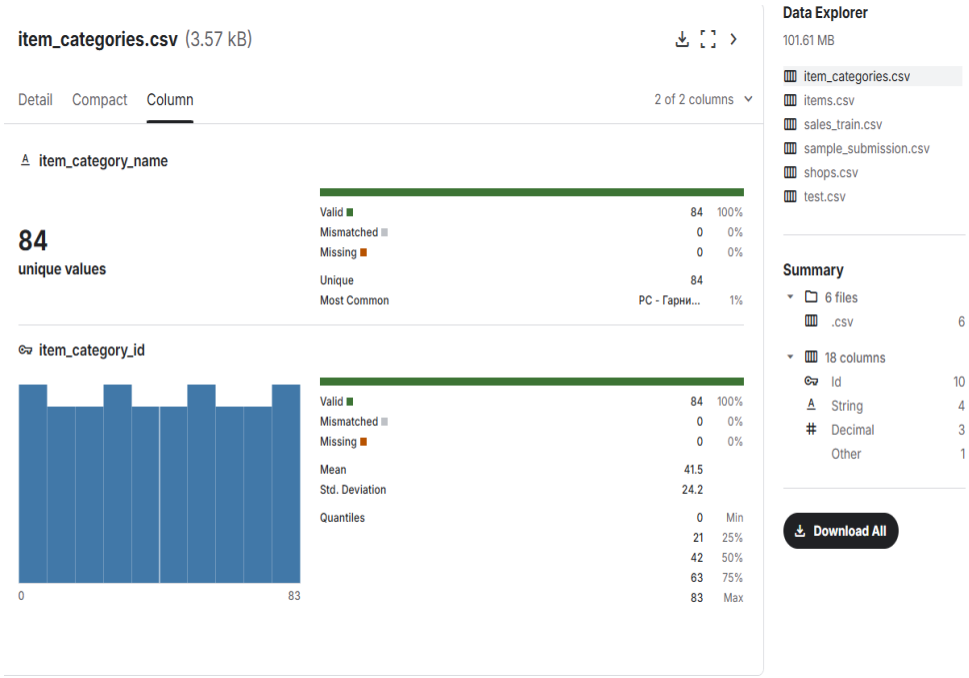
Not:

ChatGPT'yi kullanarak koduma yorumlar ekledim ve düzenledim.

Introduction:

Bu projenin amacı, geçmiş satış verilerini kullanarak mağazalar için aylık ürün satışlarını tahmin eden bir prediktif model geliştirmektir. Bu görev, büyük veri setlerinin işlenmesi, verilerin temizlenmesi ve ön işlenmesi, özellik mühendisliği, makine öğrenmesi modellerinin eğitilmesi ve performanslarının değerlendirilmesini içermektedir.

Buradaki sorun, çeşitli mağazalar ve ürünler için aylık satılan ürün sayısını (item_cnt_month) tahmin etmektir. Sağlanan veri seti, satış verileri, ürün bilgileri, mağaza detayları ve tahmin için test verilerini içermektedir.



Veri Seti Açıklaması

Proje kapsamında kullanılan veri setleri şunlardır:

- **sales_train.csv: Eğitim seti. 2013 Ocak'tan 2015 Ekim'e kadar olan günlük tarihsel verileri içerir.**
- **test.csv: Test seti**
- **sample_submission.csv: Doğru formatta örnek bir gönderim dosyasıdır.**
- **items.csv: Ürünler/hizmetler hakkında ek bilgi içerir.**
- **item_categories.csv: Ürün kategorileri hakkında ek bilgi içerir.**
- **shops.csv: Mağazalar hakkında ek bilgi içerir.**

Importing Libraries:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
import xgboost as xgb
```

pandas: Veri manipülasyonu ve analizi için kullanılır (örneğin, veri yükleme, temizleme, dönüştürme vb.).

numpy: Sayısal işlemler için kullanılır, örneğin dizilerle çalışmak ve matematiksel fonksiyonlar.

matplotlib.pyplot: Statik, animasyonlu ve etkileşimli görselleştirmeler oluşturmak için kullanılır.

```python
# Load the datasets
item_categories = pd.read_csv(r'Kaggle DataSet/item_categories.csv')
items = pd.read_csv(r'Kaggle DataSet/items.csv')
sales_train = pd.read_csv(r'Kaggle DataSet/sales_train.csv')
shops = pd.read_csv(r'Kaggle DataSet/shops.csv')
test = pd.read_csv(r'Kaggle DataSet/test.csv')
```

sklearn.model_selection.train_test_split: Veri setini eğitim ve test setlerine ayırmak için kullanılır.

sklearn.preprocessing.LabelEncoder: Kategorik değişkenleri sayısal etiketlere dönüştürmek için kullanılır.

sklearn.metrics.mean_squared_error: Ortalama Kare Hata (MSE) hesaplamak için kullanılır, bu regresyon problemleri için yaygın bir metriktir.

xgboost: Yapılandırılmış/tablolarda verimli olan popüler bir makine öğrenmesi algoritmasıdır (XGBoost, bir gradyan yükseltme çerçevesidir).

```python

# Preprocessing

sales_train = sales_train.fillna(0)
test = test.fillna(0)

```

Veri Ön İşleme
Eksik Değerlerin İşlenmesi:

Veri setlerinde eksik değerler kontrol edilir. Eğer eksik değerler bulunursa, bunlar sıfırla doldurulur (fillna(0)) ve böylece eğitim sürecinde herhangi bir kesinti yaşanmaz.

```python

# Convert dates to datetime format
sales_train['date'] = pd.to_datetime(sales_train['date'], format='%d.%m.%Y')

# Extract year, month, and day from the date
sales_train['year'] = sales_train['date'].dt.year
sales_train['month'] = sales_train['date'].dt.month
sales_train['day'] = sales_train['date'].dt.day

# Aggregating sales data by shop_id, item_id, and month
monthly_sales = sales_train.groupby(['shop_id', 'item_id', 'year', 'month'])['item_cnt_day'].sum().reset_index()

# Merge item and shop info to the aggregated sales data
monthly_sales = pd.merge(monthly_sales, items[['item_id', 'item_category_id']], on='item_id', how='left')
monthly_sales = pd.merge(monthly_sales, shops[['shop_id', 'shop_name']], on='shop_id', how='left')

# Prepare features (X) and target (y)
X = monthly_sales[['shop_id', 'item_id', 'item_category_id', 'year', 'month']]
y = monthly_sales['item_cnt_day']

# Encoding categorical features with LabelEncoder
label_encoder = LabelEncoder()
X_encoded = X.copy()

# Apply LabelEncoder on each categorical column to avoid SettingWithCopyWarning
X_encoded['shop_id'] = label_encoder.fit_transform(X_encoded['shop_id'])
X_encoded['item_id'] = label_encoder.fit_transform(X_encoded['item_id'])
X_encoded['item_category_id'] = label_encoder.fit_transform(X_encoded['item_category_id'])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split( *arrays: X_encoded, y, test_size=0.2, random_state=42)

print(f"Train size: {X_train.shape}, Test size: {X_test.shape}")

```

Kod, satış verilerini makine öğrenmesi analizine hazırlamak için şu adımları izler:

1. **Tarih Kolonunun Formatlanması**: Tarih sütunu, yıl, ay ve gün gibi ilgili bileşenleri çıkartmak için datetime formatına dönüştürülür.
2. **Aylık Satışların Toplanması**: Satış verileri, mağaza, ürün, yıl ve ay bazında gruplanarak her kombinasyon için toplam satışlar hesaplanır.
3. **Veri Zenginleştirme**: Toplanan veriler, ürün kategorileri ve mağaza adları gibi ek bilgilerle birleştirilir.
4. **Özelliklerin Ayrılması**: Makine öğrenmesi için kullanılacak özellikler (shop ID, item ID, item category ID, yıl, ay) hedef değişken olan toplam satış (item_cnt_day) ile ayrılır.
5. **Kategorik Değişkenlerin Kodlanması**: Kategorik değişkenler, makine öğrenmesi modellerine uygun hale getirilmesi için sayısal değerlere dönüştürülür (Label Encoding kullanılarak).
6. **Eğitim ve Test Setlerine Ayırma**: Veri, %80 eğitim ve %20 test olmak üzere ikiye ayrılır, böylece model doğru şekilde eğitilip değerlendirilebilir.

Bu süreç, verilerin model için uygun hale getirilmesini sağlar ve doğru tahminler elde etmek için gereklidir.

```
60
61  # Train an XGBoost regressor
62  model = xgb.XGBRegressor(objective='reg:squarederror', colsample_bytree=0.3, learning_rate=0.1,
63                           max_depth=5, alpha=10, n_estimators=100)
64
65  model.fit(X_train, y_train)
66
67  # Make predictions on the test set
68  y_pred = model.predict(X_test)
69
70  # Evaluate the model
71  rmse = np.sqrt(mean_squared_error(y_test, y_pred))
72  print(f"Root Mean Squared Error (RMSE): {rmse}")
```

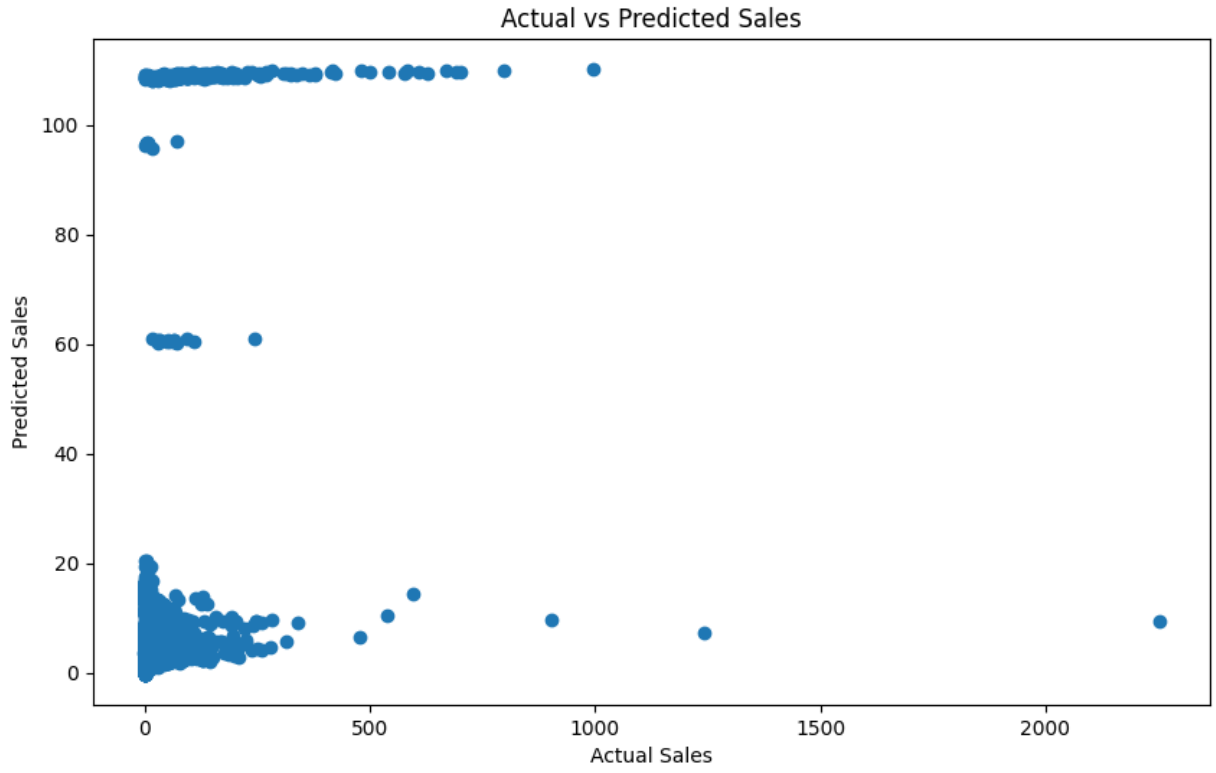
Bu kod parçasında, satış verilerini tahmin etmek için bir XGBoost regresyon modeli eğitilmektedir. İşlem şu adımları içerir:

1. **Modelin Başlatılması**:
   İlk olarak, XGBRegressor belirli hiperparametrelerle başlatılır:
   - objective='reg:squarederror': Bu, modelin bir regresyon görevi olduğunu ve kayıp fonksiyonu olarak kare hatayı (squared error) kullandığını belirtir.
   - Diğer hiperparametreler:
     - colsample_bytree=0.3: Her bir ağacın eğitiminde kullanılan örneklerin yüzde 30'unu seçer.
     - learning_rate=0.1: Modelin öğrenme oranını belirler, bu da modelin her iterasyondaki adım büyüklüğünü kontrol eder.
     - max_depth=5: Her ağacın maksimum derinliğini sınırlar, bu da modelin karmaşıklığını kontrol eder.
     - alpha=10: L2 düzenlileme parametresi, modelin aşırı öğrenmesini engellemeye yardımcı olur.
     - n_estimators=100: Boosting adımlarının sayısını belirler, yani modelin kaç kez iyileştirileceğini.
2. **Modelin Eğitilmesi**:
   Model, fit yöntemi kullanılarak eğitim verileriyle (X_train ve y_train) eğitilir.
3. **Tahminlerin Yapılması**:
   Model eğitildikten sonra, test seti (X_test) üzerinde tahminler yapılır. Bu tahminler, predict yöntemiyle elde edilir.
4. **Modelin Değerlendirilmesi**:
   Modelin performansını değerlendirmek için, tahmin edilen ve gerçek değerler arasındaki **Root Mean Squared Error (RMSE)** hesaplanır. RMSE, regresyon görevleri için yaygın bir değerlendirme metriğidir ve modelin satışları tahmin etme doğruluğunu gösterir. Son olarak, RMSE değeri konsola yazdırılarak modelin başarısı gösterilir.

Bu adımlar, modelin doğru şekilde eğitilmesini ve test edilmesini sağlar, böylece satış tahminleri yapılabilir ve modelin başarısı ölçülebilir.

Veri Görselleştirme

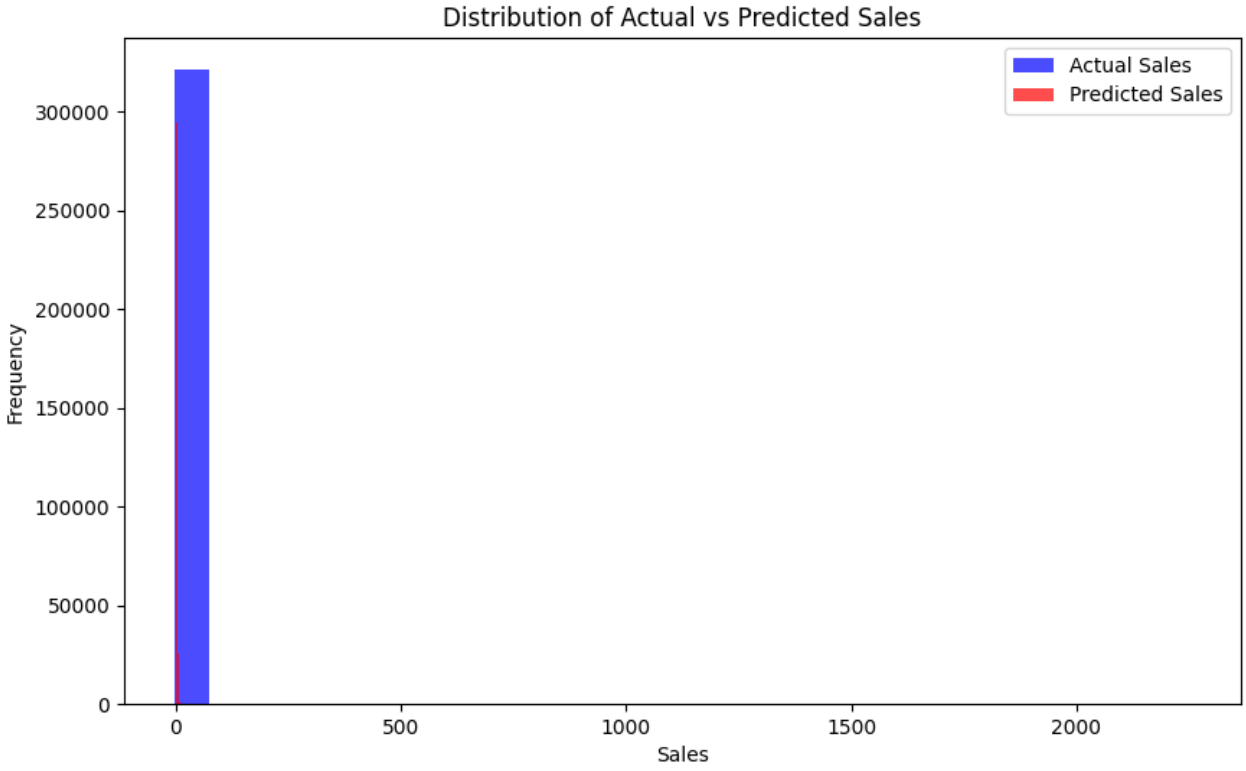
```
67  # Plotting the actual vs predicted sales
68  plt.figure(figsize=(10, 6))
69  plt.scatter(y_test, y_pred)
70  plt.xlabel("Actual Sales")
71  plt.ylabel("Predicted Sales")
72  plt.title("Actual vs Predicted Sales")
73  plt.show()
```



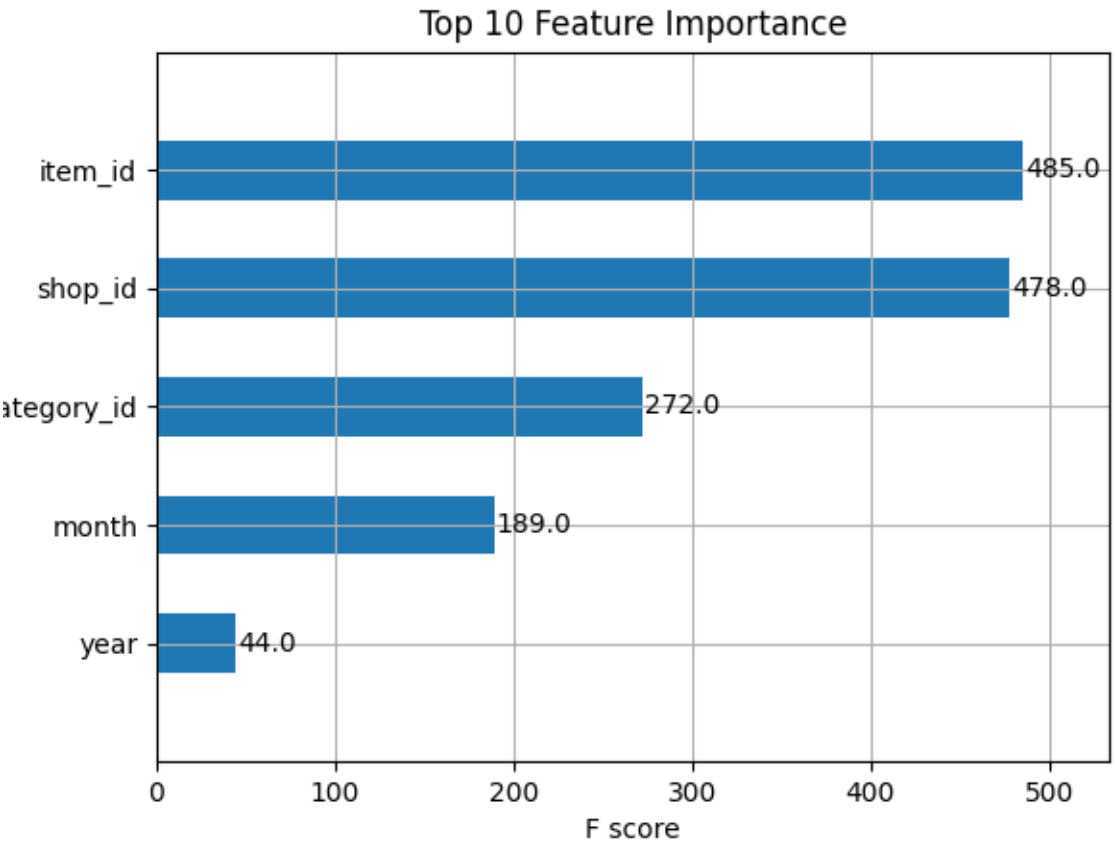
```python

# Plotting the distribution of actual sales values
plt.figure(figsize=(10, 6))
plt.hist(y_test, bins=30, alpha=0.7, color='blue', label='Actual Sales')
plt.hist(y_pred, bins=30, alpha=0.7, color='red', label='Predicted Sales')
plt.xlabel("Sales")
plt.ylabel("Frequency")
plt.title("Distribution of Actual vs Predicted Sales")
plt.legend()
plt.show()

```



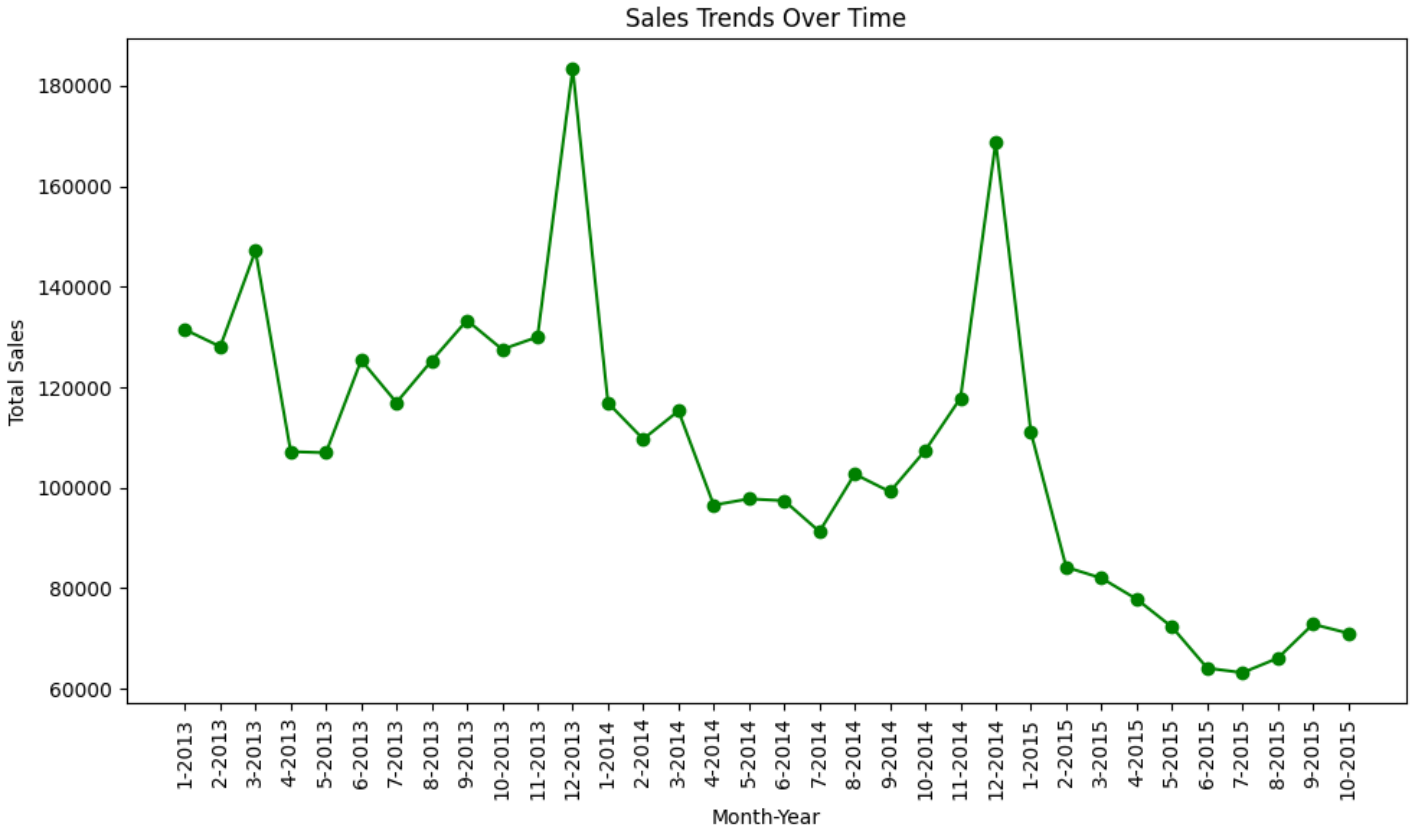
```
84
85  # Feature importance from XGBoost
86  plt.figure(figsize=(10, 6))
87  xgb.plot_importance(model, importance_type='weight', max_num_features=10, height=0.5)
88  plt.title("Top 10 Feature Importance")
89  plt.show()
90
```



```python

# Plotting the sales trends over time (by month)
monthly_sales_trends = sales_train.groupby(['year', 'month'])['item_cnt_day'].sum().reset_index()

plt.figure(figsize=(10, 6))
plt.plot( *args: monthly_sales_trends['month'].astype(str) + '-' + monthly_sales_trends['year'].astype(str),
         monthly_sales_trends['item_cnt_day'], marker='o', color='g')
plt.xticks(rotation=90)
plt.xlabel('Month-Year')
plt.ylabel('Total Sales')
plt.title('Sales Trends Over Time')
plt.tight_layout()
plt.show()
```



```
103
104 # Plotting sales by shop_id to see sales performance per shop
105 shop_sales = monthly_sales.groupby('shop_id')['item_cnt_day'].sum().reset_index()
106
107 plt.figure(figsize=(10, 6))
108 plt.bar(shop_sales['shop_id'], shop_sales['item_cnt_day'], color='orange')
109 plt.xlabel('Shop ID')
110 plt.ylabel('Total Sales')
111 plt.title('Total Sales by Shop')
112 plt.show()
113
```

KOD Tumu

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
import xgboost as xgb

# Load the datasets
item_categories = pd.read_csv(r'Kaggle DataSet/item_categories.csv')
items = pd.read_csv(r'Kaggle DataSet/items.csv')
sales_train = pd.read_csv(r'Kaggle DataSet/sales_train.csv')
shops = pd.read_csv(r'Kaggle DataSet/shops.csv')
test = pd.read_csv(r'Kaggle DataSet/test.csv')

# Preprocessing
# Handle missing values (if any)
sales_train = sales_train.fillna(0)
test = test.fillna(0)

# Convert dates to datetime format
sales_train['date'] = pd.to_datetime(sales_train['date'], format='%d.%m.%Y')

# Extract year, month, and day from the date
sales_train['year'] = sales_train['date'].dt.year
sales_train['month'] = sales_train['date'].dt.month
sales_train['day'] = sales_train['date'].dt.day

# Aggregating sales data by shop_id, item_id, and month
monthly_sales = sales_train.groupby(['shop_id', 'item_id', 'year', 'month'])['item_cnt_day'].sum().reset_index()

# Merge item and shop info to the aggregated sales data
monthly_sales = pd.merge(monthly_sales, items[['item_id', 'item_category_id']], on='item_id', how='left')
monthly_sales = pd.merge(monthly_sales, shops[['shop_id', 'shop_name']], on='shop_id', how='left')

# Prepare features (X) and target (y)
X = monthly_sales[['shop_id', 'item_id', 'item_category_id', 'year', 'month']]
y = monthly_sales['item_cnt_day']

# Encoding categorical features with LabelEncoder
label_encoder = LabelEncoder()
X_encoded = X.copy()

# Apply LabelEncoder on each categorical column to avoid SettingWithCopyWarning
X_encoded['shop_id'] = label_encoder.fit_transform(X_encoded['shop_id'])
X_encoded['item_id'] = label_encoder.fit_transform(X_encoded['item_id'])
X_encoded['item_category_id'] = label_encoder.fit_transform(X_encoded['item_category_id'])
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

print(f"Train size: {X_train.shape}, Test size: {X_test.shape}")

# Train an XGBoost regressor
model = xgb.XGBRegressor(objective='reg:squarederror', colsample_bytree=0.3, learning_rate=0.1,
                max_depth=5, alpha=10, n_estimators=100)

model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Plotting the actual vs predicted sales
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.title("Actual vs Predicted Sales")
plt.show()

# Plotting the distribution of actual sales values
plt.figure(figsize=(10, 6))
plt.hist(y_test, bins=30, alpha=0.7, color='blue', label='Actual Sales')
plt.hist(y_pred, bins=30, alpha=0.7, color='red', label='Predicted Sales')
plt.xlabel("Sales")
plt.ylabel("Frequency")
plt.title("Distribution of Actual vs Predicted Sales")
plt.legend()
plt.show()

# Feature importance from XGBoost
plt.figure(figsize=(10, 6))
xgb.plot_importance(model, importance_type='weight', max_num_features=10, height=0.5)
plt.title("Top 10 Feature Importance")
plt.show()
```

```
# Plotting the sales trends over time (by month)
monthly_sales_trends = sales_train.groupby(['year', 'month'])['item_cnt_day'].sum().reset_index()

plt.figure(figsize=(10, 6))
plt.plot(monthly_sales_trends['month'].astype(str) + '-' + monthly_sales_trends['year'].astype(str),
      monthly_sales_trends['item_cnt_day'], marker='o', color='g')
plt.xticks(rotation=90)
plt.xlabel('Month-Year')
plt.ylabel('Total Sales')
plt.title('Sales Trends Over Time')
plt.tight_layout()
plt.show()

# Plotting sales by shop_id to see sales performance per shop
shop_sales = monthly_sales.groupby('shop_id')['item_cnt_day'].sum().reset_index()

plt.figure(figsize=(10, 6))
plt.bar(shop_sales['shop_id'], shop_sales['item_cnt_day'], color='orange')
plt.xlabel('Shop ID')
plt.ylabel('Total Sales')
plt.title('Total Sales by Shop')
plt.show()

print("bitti")
```