KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YÜKSEK DÜZEY PROGRAMLAMA DERSİ

PREDICT FUTURE SALES

NARİN ULUIŞIK
202113171017

1.Giriş

Perakende sektöründe geçmiş satış verilerini kullanarak gelecekteki satışları tahmin etmek, karar alma süreçlerini iyileştirmek açısından kritik öneme sahiptir. Bu projede, günlük satış adetlerini tahmin etmek için XGBoost algoritması kullanılmıştır. Veri setleri üzerinde ön işleme, model eğitimi ve performans analizi yapılmıştır.

2. Amaç ve Kapsam

Bu projenin temel amacı, geçmiş satış verilerinden hareketle, mağaza ve ürün bazında gelecekteki satış adetlerini tahmin edebilen bir makine öğrenimi modeli geliştirmektir. Bu kapsamda aşağıdaki adımlar gerçekleştirilmiştir:

- Veri setlerinin yüklenmesi ve incelenmesi.
- Veri ön işleme süreçlerinin uygulanması.
- Tahmin modeli oluşturma ve değerlendirme.
- Gerçek ve tahmin edilen değerlerin karşılaştırılması.

3.Kullanılan Kütüphaneler

Çalışmada aşağıdaki Python kütüphaneleri kullanılmıştır:

```
In [2]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from xgboost import XGBRegressor
```

4.Veri Setlerinin Yüklenmesi

Projede kullanılan veri setleri, satış verileri ve ürün bilgilerini içermektedir. Aşağıda, her bir veri setinin yüklenmesi ve ilk birkaç satırının incelenmesi için kullanılan Python kodu bulunmaktadır:

```
In [3]: # Veri setlerini yükleyelim
        sales_train = pd.read_csv('sales_train.csv')
        items = pd.read_csv('items.csv')
        item_categories = pd.read_csv('item_categories.csv')
        shops = pd.read_csv('shops.csv')
        test = pd.read_csv('test.csv')

        # Verinin ilk birkaç satırını inceleyelim
        print(sales_train.head())
        print(items.head())
        print(item_categories.head())
        print(shops.head())
        print(test.head())
```

```
         date  date_block_num  shop_id  item_id  item_price  item_cnt_day
0  02.01.2013               0       59    22154      999.00           1.0
1  03.01.2013               0       25     2552      899.00           1.0
2  05.01.2013               0       25     2552      899.00          -1.0
3  06.01.2013               0       25     2554     1709.05           1.0
4  15.01.2013               0       25     2555     1099.00           1.0

                                           item_name  item_id  \
0          ! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.)         D        0
1  !ABBYY FineReader 12 Professional Edition Full...        1
2     ***В ЛУЧАХ СЛАВЫ   (UNV)                    D        2
3    ***ГОЛУБАЯ ВОЛНА  (Univ)                     D        3
4        ***КОРОБКА (СТЕКЛО)                      D        4

   item_category_id
0                40
1                76
2                40
3                40
4                40
```

5.Verilerin Birleştirilmesi ve İşlenmesi

Bu aşamada, farklı veri setlerinden elde edilen bilgiler tek bir veri setinde birleştirilmiş ve bazı tarihsel veriler türetilmiştir.

```
In [4]: # Verileri birleştirelim
sales_data = sales_train.merge(items, on='item_id', how='left')
sales_data = sales_data.merge(item_categories, on='item_category_id', how='left')
sales_data = sales_data.merge(shops, on='shop_id', how='left')

# 'date' sütununu datetime formatına çeviriyoruz
sales_data['date'] = pd.to_datetime(sales_data['date'], format='%d.%m.%Y')

# Yıl, ay, gün, haftanın günü gibi bilgileri türetiyoruz
sales_data['year'] = sales_data['date'].dt.year
sales_data['month'] = sales_data['date'].dt.month
sales_data['day'] = sales_data['date'].dt.day
sales_data['weekday'] = sales_data['date'].dt.weekday  # 0: Pazartesi, 6: Pazar

# Verinin ilk 5 satırını kontrol edelim
print(sales_data.head())
```

```
        date  date_block_num  shop_id  item_id  item_price  item_cnt_day  \
0 2013-01-02               0       59    22154      999.00           1.0
1 2013-01-03               0       25     2552      899.00           1.0
2 2013-01-05               0       25     2552      899.00          -1.0
3 2013-01-06               0       25     2554     1709.05           1.0
4 2013-01-15               0       25     2555     1099.00           1.0

                            item_name  item_category_id  \
0                      ЯВЛЕНИЕ 2012 (BD)                37
1  DEEP PURPLE  The House Of Blue Light  LP                58
2  DEEP PURPLE  The House Of Blue Light  LP                58
3  DEEP PURPLE  Who Do You Think We Are  LP                58
4   DEEP PURPLE 30 Very Best Of 2CD (Фирм.)                56

                   item_category_name              shop_name  year  month  \
0                    Кино - Blu-Ray  Ярославль ТЦ "Альтаир"  2013      1
1                    Музыка - Винил     Москва ТРК "Атриум"  2013      1
2                    Музыка - Винил     Москва ТРК "Атриум"  2013      1
3                    Музыка - Винил     Москва ТРК "Атриум"  2013      1
4 Музыка - CD фирменного производства     Москва ТРК "Атриум"  2013      1
```

- **Veri Birleştirme:** sales_train, items, item_categories, ve shops veri setleri, ortak alanlarla (item_id, item_category_id, shop_id) birleştirilmiştir.
- **Tarih İşleme:** date sütunu, tarih formatına dönüştürülüp, yıl, ay, gün ve haftanın günü bilgileri türetilmiştir.

6.Eğitim ve Test Setine Ayırma

Bu adımda, veriyi eğitim ve test setlerine ayırıyoruz. Test seti, modelin doğruluğunu değerlendirmek için kullanılacak.

- **train_test_split**: Veriyi eğitim ve test setlerine böler. Burada, verinin %80'i eğitim için, %20'si ise test için ayrılır.
- **random_state=42**: Veri bölme işleminin tekrar edilebilir olmasını sağlar

```
from sklearn.model_selection import train_test_split

# Eğitim ve test setlerine ayıralım
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Eğitim ve test setlerinin boyutlarını kontrol edelim
print(X_train.shape, X_test.shape)
```

```
(2348679, 7) (587170, 7)
```

7.Modelin Eğitilmesi ve Tahmin Yapılması

Bu adımda, XGBoost regresyon modelini kullanarak eğitim setiyle model eğitilir ve test seti üzerinde tahminler yapılır.

- **XGBRegressor**: XGBoost algoritmasını regresyon problemleri için kullanır. Burada objective='reg:squarederror' ile modelin hata fonksiyonu olarak kare hata kullanılacağı belirtilir.
- **fit()**: Model, eğitim verisiyle eğitilir.
- **predict()**: Eğitilen model, test verisi üzerinde tahminler yapar

```
In [8]: from xgboost import XGBRegressor

        # XGBoost modelini oluşturuyoruz
        model = XGBRegressor(objective='reg:squarederror', random_state=42)

        # Modeli eğitelim
        model.fit(X_train, y_train)

        # Eğitim sonrası modelin tahmin yapabilmesi için hazır hale gelmesini s
        y_pred = model.predict(X_test)

        # Tahmin sonuçlarını inceleyelim
        print(y_pred[:5])

        [1.2585999 1.0840536 1.3041766 1.199165  1.008466 ]
```

8.Model Değerlendirmesi

Modelin başarısını değerlendirmek için, test seti üzerinde yapılan tahminlerin doğruluğunu ölçmek amacıyla **Root Mean Squared Error (RMSE)** hesaplanır.

- **mean_squared_error**: Gerçek değerler (y_test) ile modelin tahminleri (y_pred) arasındaki ortalama kare hata (MSE) hesaplanır.
- **math.sqrt**: MSE'nin karekökü alınarak RMSE (Root Mean Squared Error) hesaplanır. RMSE, modelin hata büyüklüğünü anlamak için yaygın olarak kullanılan bir metriktir.

```
In [9]: from sklearn.metrics import mean_squared_error
        import math

        # Hata hesaplaması
        mse = mean_squared_error(y_test, y_pred)
        rmse = math.sqrt(mse)

        # RMSE sonucunu yazdıralım
        print(f'Root Mean Squared Error (RMSE): {rmse}')

        Root Mean Squared Error (RMSE): 1.7983591858562573
```

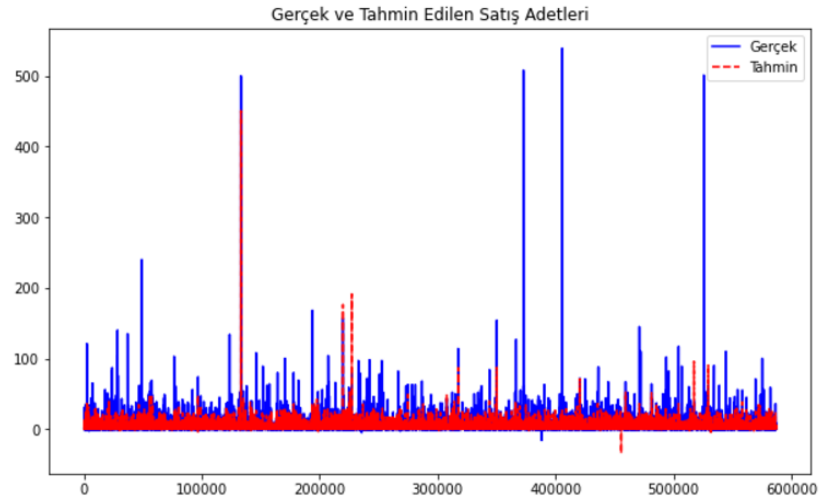
9.Gerçek ve Tahmin Edilen Satış Adetleri Görselleştirmesi

Modelin tahminlerinin görsel olarak değerlendirilmesi için, gerçek ve tahmin edilen satış adetlerini karşılaştıran bir grafik çizilir. Bu, modelin performansını anlamamıza yardımcı olur.

- **plt.plot**: Gerçek ve tahmin edilen değerler çizilir. Gerçek değerler mavi, tahmin edilen değerler ise kırmızı renkte gösterilir.
- **label**: Her çizgiye bir etiket eklenir (Gerçek ve Tahmin).
- **legend**: Grafik üzerinde etiketleri gösteren bir açıklama ekler.
- **title**: Grafiğin başlığını belirtir.

In [10]:
```
import matplotlib.pyplot as plt

# Gerçek ve tahmin edilen değerleri görselleştirelim
plt.figure(figsize=(10, 6))
plt.plot(y_test.values, label='Gerçek', color='blue')
plt.plot(y_pred, label='Tahmin', color='red', linestyle='dashed')
plt.legend()
plt.title('Gerçek ve Tahmin Edilen Satış Adetleri')
plt.show()
```



10.Sonuç ve Değerlendirme

Bu proje kapsamında XGBoost algoritması ile günlük satış adetleri tahmin edilmiştir. Tahmin sonuçları genel olarak gerçek değerlere yakın bulunmuş, RMSE değeri ile model performansı ölçülmüştür. Projenin ilerleyen aşamalarında daha fazla özellik eklenerek modelin doğruluğu artırılabilir.