KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ

YÜKSEK DÜZEY PROGRAMLAMA PROJE RAPORU
KONU : PREDICT FUTURE SALES

HAZIRLAYAN : 202013172006 MUHAMMET ALİ BAL
DANIŞMAN : DOÇ. DR. HASAN TEMURTAŞ

KODLAR :

```
# Gerekli Kütüphaneleri Yükleme
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error
import xgboost as xgb
```

1. ***`pandas`:*** *Veri manipülasyonu ve analiz işlemleri için kullanılan, veri yapıları (DataFrame, Series) sunan bir kütüphanedir. Veri setlerini yüklemek, birleştirmek ve ön işleme yapmak için kullanılır.*

2. ***`numpy`:*** *Matematiksel hesaplamalar ve veri işleme için kullanılan bir kütüphanedir. Genellikle diziler (arrays) üzerinde hızlı hesaplamalar yapmak için kullanılır.*

3. ***`matplotlib.pyplot`:*** *Veri görselleştirme için kullanılan bir kütüphanedir. Grafikler, çizgi ve bar grafikler gibi görseller oluşturmak için kullanılır.*

4. ***`seaborn`:*** *Matplotlib üzerine inşa edilmiş, veri görselleştirmeyi kolaylaştıran bir kütüphanedir. Estetik ve daha zengin görselleştirmeler için kullanılır.*

5. ***`sklearn.model_selection`:*** *Makine öğrenmesi modellerinin eğitimi ve hiperparametre optimizasyonu için kullanılan araçları içerir.* ***`train_test_split`*** *veri setini eğitim ve test kümelerine ayırırken,* ***`GridSearchCV`*** *hiperparametre optimizasyonu için kullanılır.*

6. ***`sklearn.metrics`:*** *Model değerlendirmesi için kullanılan metrikleri içerir.* ***`mean_squared_error`*** *modeli değerlendirmek için kullanılır.*

7. ***`xgboost`:*** *Yüksek performanslı, karar ağaçları tabanlı bir modelleme algoritmasıdır. Bu kütüphane, regresyon ve sınıflandırma problemlerinde güçlü sonuçlar elde etmek için kullanılır.*

```
# Veri Setlerinin Yüklenmesi
item_categories = pd.read_csv('item_categories.csv')
items = pd.read_csv('items.csv')
shops = pd.read_csv('shops.csv')
train = pd.read_csv('sales_train.csv')
test = pd.read_csv('test.csv')
```

1. ***`item_categories.csv`:*** *Ürün kategorilerini içeren veri seti. Her ürünün ait olduğu kategori bilgilerini içerir.*

2. ***`items.csv`:*** *Ürünlere ait bilgileri içerir. Ürünlerin ID'leri ve diğer özellikleri (örneğin adı, kategorisi) bu dosyada bulunur.*

3. ***`shops.csv`:*** *Mağazalarla ilgili bilgileri içerir. Her mağazanın ID'si ve adı gibi bilgileri barındırır.*

4. ***`sales_train.csv`:*** *Eğitim veri seti. Geçmişte yapılan satışlara dair verileri içerir. Her satırda satış tarihi, ürün ID'si, mağaza ID'si, satılan miktar ve fiyat bilgileri yer alır.*

5. ***`test.csv`:*** *Test veri seti. Bu veri, modelin tahmin yapması gereken test verilerini içerir. Gerçek satış verisi bulunmaz, ancak modelin tahmin yapacağı mağaza ve ürün bilgileri içerir.*

```
# Verilerin Birleştirilmesi
train = train.merge(items, on='item_id', how='left')
train = train.merge(item_categories, on='item_category_id', how='left')
test = test.merge(items, on='item_id', how='left')
test = test.merge(item_categories, on='item_category_id', how='left')

# Tarih Formatının Dönüştürülmesi
train['date'] = pd.to_datetime(train['date'], format='%d.%m.%Y')
train['year'] = train['date'].dt.year
train['month'] = train['date'].dt.month

# Toplam Satış Sütununun Eklenmesi
train['total_sales'] = train['item_price'] * train['item_cnt_day']
```

Verilerin Birleştirilmesi :

- ***Eğitim (`train`) ve test (`test`) veri setlerine, ürünler (`items`) ve ürün kategorileri (`item_categories`) ile ilişkilendirilmiş bilgileri eklemek için `merge()` fonksiyonu kullanılarak birleştirilmiştir. Bu işlemle, her bir satışa karşılık gelen ürün adı ve kategori bilgisi eklenmiştir.***

Tarih Formatının Dönüştürülmesi :

- ***`train` veri setindeki tarih sütunu, `pd.to_datetime()` fonksiyonu ile datetime formatına dönüştürülüp, yıl (`year`) ve ay (`month`) bilgileri ayrı sütunlar olarak eklenmiştir.***

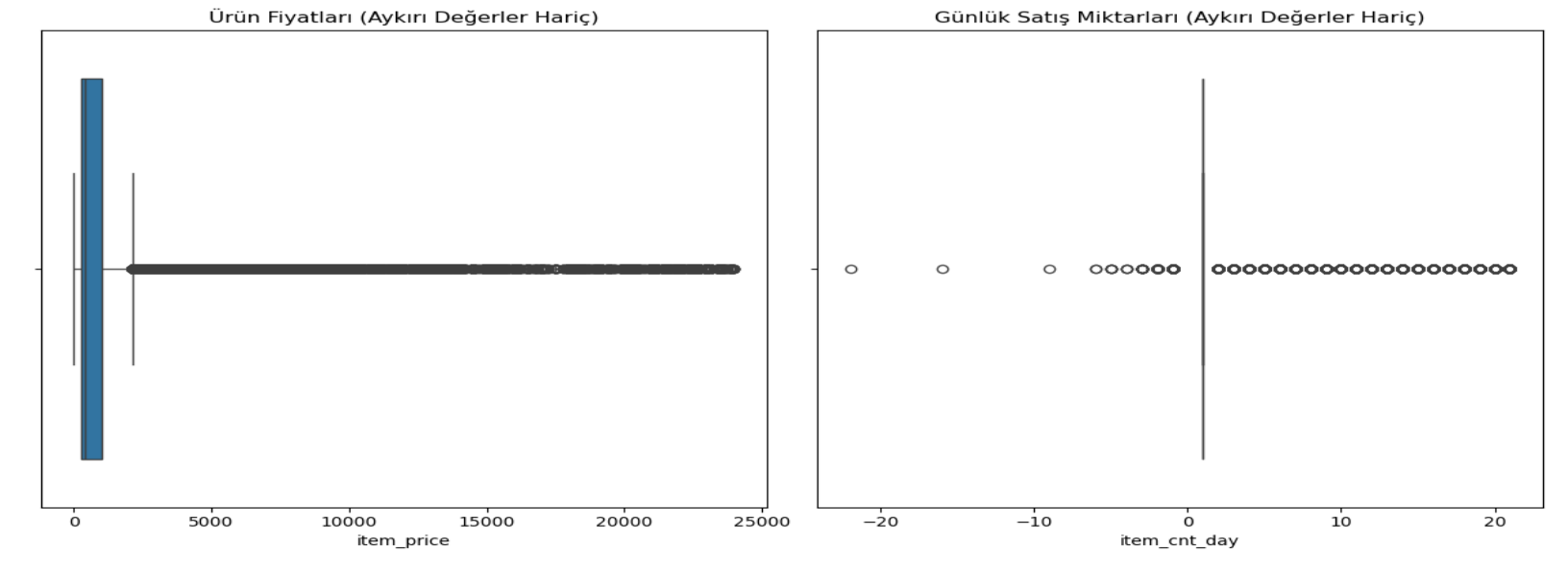
Toplam Satış Sütununun Eklenmesi :

- ***Her satışın toplam değerini hesaplamak için `total_sales` sütunu eklenmiş, bu sütun ürün fiyatı (`item_price`) ile satılan ürün adedi (`item_cnt_day`) çarpılarak oluşturulmuştur.***

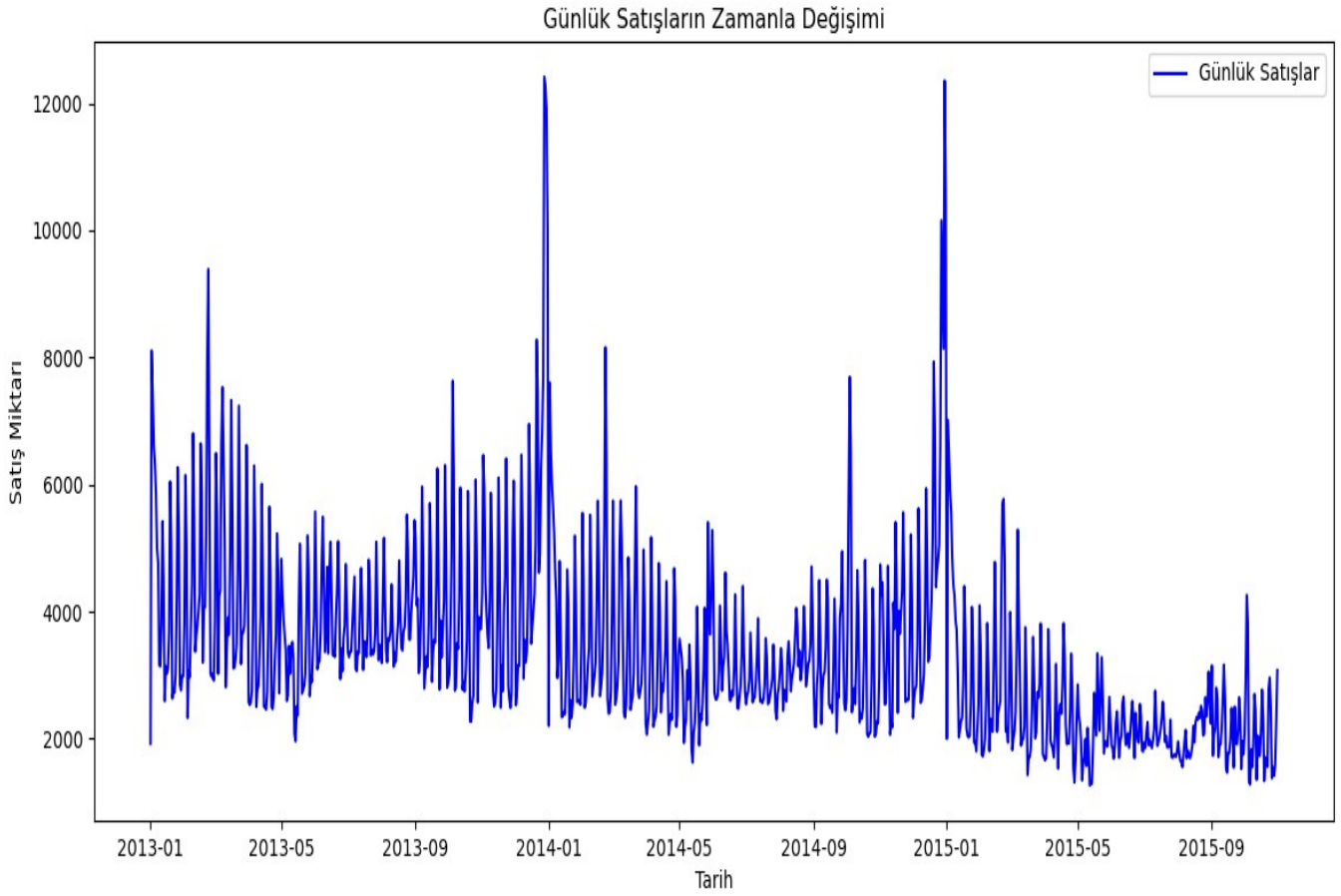
```
# Aykırı Değerlerin Temizlenmesi
train = train[(train['item_price'] > 0) &
              (train['item_price'] < train['item_price'].quantile(0.999))]
train = train[(train['item_cnt_day'] < train['item_cnt_day'].quantile(0.999))]

# Aykırı Değerlerin Kutu Grafikleri
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.boxplot(x=train['item_price'])
plt.title("Ürün Fiyatları (Aykırı Değerler Hariç)")
plt.subplot(1, 2, 2)
sns.boxplot(x=train['item_cnt_day'])
plt.title("Günlük Satış Miktarları (Aykırı Değerler Hariç)")
plt.tight_layout()
plt.show()

# Zaman Serisi Analizi
sales_daily = train.groupby('date')['item_cnt_day'].sum()
```



```python
# Günlük Satışların Trendi
plt.figure(figsize=(14, 6))
plt.plot(sales_daily, label='Günlük Satışlar', color='blue')
plt.title("Günlük Satışların Zamanla Değişimi")
plt.xlabel("Tarih")
plt.ylabel("Satış Miktarı")
plt.legend()
plt.show()
```



```
# Özellik Ataması
monthly_sales = train.groupby(['date_block_num', 'shop_id'],
                              as_index=False).agg({'total_sales': 'sum'})

# Özelliklerin Ayrılması
X = monthly_sales[['date_block_num', 'shop_id']]  # Girdi Özellikleri
y = monthly_sales['total_sales']                  # Hedef Değişken
```

Özellik Ataması :

`train` veri seti, her mağaza (`shop_id`) ve her zaman bloğu (`date_block_num`) için toplam satışları hesaplamak amacıyla gruplanmıştır. Bu işlemle, her mağaza ve zaman bloğu için toplam satış miktarları (`total_sales`) bulunmuş ve `monthly_sales` veri seti oluşturulmuştur.

Özelliklerin Ayrılması :

`X` olarak, modelin girdi özellikleri seçilmiştir: `date_block_num` (zaman bloğu) ve `shop_id` (mağaza ID'si).
`y` olarak ise, hedef değişken olan `total_sales` (toplam satış) seçilmiştir. Bu sütun, modelin tahmin etmeye çalışacağı değeri temsil eder.

```
# Eğitim ve Test Verilerinin Ayrılması
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)

# XGBoost Modeli için GridSearchCV ile Hiperparametre Optimizasyonu
param_grid = {
    'n_estimators': [50, 100],
    'learning_rate': [0.01, 0.2],
    'max_depth': [3, 5],
    'subsample': [0.7, 1.0],
    'colsample_bytree': [0.7, 1.0],
    'gamma': [0, 0.2]
}

xgb_model = xgb.XGBRegressor()
grid_search = GridSearchCV(estimator=xgb_model,
                           param_grid=param_grid,
                           scoring='neg_mean_squared_error',
                           cv=5,
                           verbose=1,
                           n_jobs=-1)
grid_search.fit(X_train, y_train)
```

Eğitim ve Test Verilerinin Ayrılması :

`train_test_split()` fonksiyonu ile, veriler eğitim (`X_train`, `y_train`) ve test (`X_test`, `y_test`) setlerine ayrılmıştır. Bu işlem, modelin eğitim ve test aşamalarında kullanacağı verilerin %80'ini eğitim, %20'sini ise test verisi olarak ayırmıştır.

XGBoost Modeli için GridSearchCV ile Hiperparametre Optimizasyonu :

- ***XGBoost regresyon modelinin performansını artırmak için `GridSearchCV` kullanılmıştır. `param_grid` ile modelin hiperparametreleri belirlenmiş ve bu parametrelerin farklı kombinasyonları üzerinde çapraz doğrulama yapılmıştır. Bu işlem, en iyi model parametrelerini seçmek amacıyla kullanılır ve modelin daha iyi performans göstermesini sağlar.***

```python
# En İyi Parametrelerin Seçilmesi
best_model = grid_search.best_estimator_

# Model Performansının Değerlendirilmesi
y_pred = best_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Validation RMSE:", rmse)
```

```
Validation RMSE: 567852.5657267721
```

En İyi Parametrelerin Seçilmesi :

`grid_search.best_estimator_` ile, hiperparametre optimizasyonu sonucunda elde edilen en iyi model (`best_model`) seçilmiştir.

Model Performansının Değerlendirilmesi :

`best_model` ile test verisi (`X_test`) üzerinde tahminler yapılmış ve bu tahminler ile gerçek test değerleri (`y_test`) karşılaştırılmıştır.
Modelin performansı, `mean_squared_error` (ortalama kare hata) fonksiyonu ile hesaplanmış ve bu hata değeri karekköküne alınarak Root Mean Squared Error (RMSE) değeri bulunmuştur. RMSE, modelin tahminlerinin ne kadar doğru olduğunu gösteren bir metriktir.

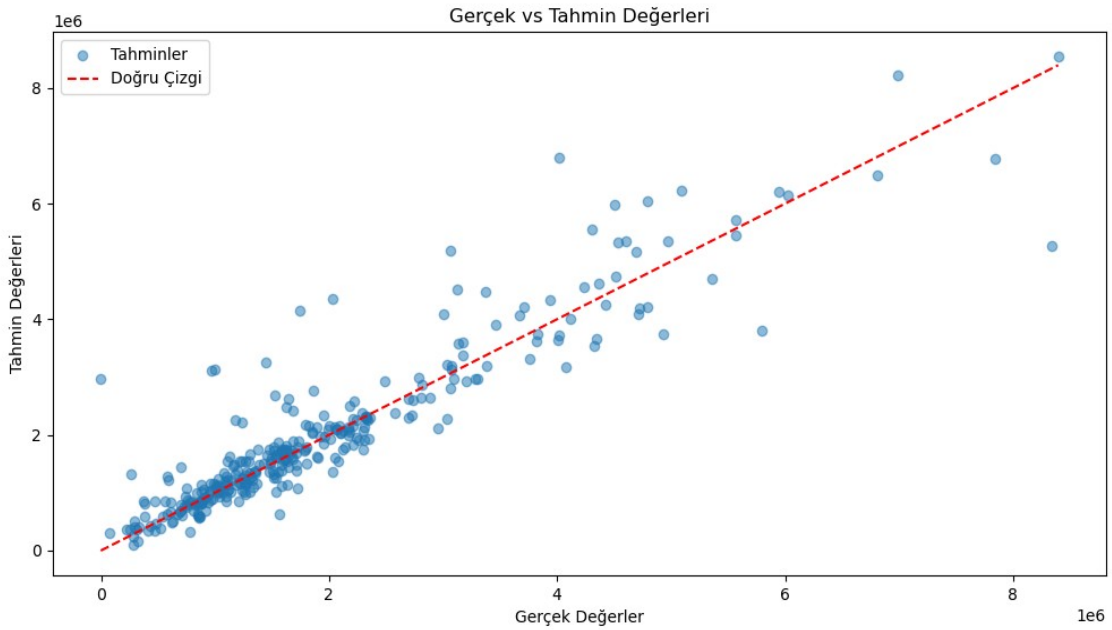
```python
# Model Doğruluk Grafiği
plt.figure(figsize=(12, 6))
plt.scatter(y_test, y_pred, alpha=0.5, label='Tahminler')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
         'r--', label='Doğru Çizgi')
plt.title("Gerçek vs Tahmin Değerleri")
plt.xlabel("Gerçek Değerler")
plt.ylabel("Tahmin Değerleri")
plt.legend()
plt.show()

# Test Verisiyle Tahmin Yapılması
test['date_block_num'] = max(train['date_block_num']) + 1
test_X = test[['date_block_num', 'shop_id']]
test['item_cnt_month'] = best_model.predict(test_X)

# Tahmin Sonuçlarının Kaydedilmesi
submission = test[['ID', 'item_cnt_month']]
submission.to_csv('submission.csv', index=False)

print("Tahminler submission.csv dosyasına kaydedildi.")
```

```
Tahminler submission.csv dosyasına kaydedildi.
```



submission - DataFrame

Index	ID	tem_cnt_month
0	0	1042429.7
1	1	1042429.7
2	2	1042429.7
3	3	1042429.7
4	4	1042429.7
5	5	1042429.7
6	6	1042429.7
7	7	1042429.7
8	8	1042429.7
9	9	1042429.7
10	10	1042429.7