```
from google.colab import drive
drive.mount('/content/drive/')
```

```
import os
os.chdir('/content/drive/My Drive/MakineSon')
!pwd
```

```
!pip install xgboost
```

```
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

```
from warnings import filterwarnings
filterwarnings("ignore")
```

```
df_2 = pd.read_csv(r"istanbulson2.csv")  # 'istanbulson2.csv'
adlı CSV dosyasını oku ve df_2 adlı DataFrame'e yükle
df = df_2.copy()  # df_2'nin bir kopyasını oluştur ve df adlı
yeni bir DataFrame'e ata
```

```
X = df.drop(["fiyat"], axis=1)  # 'fiyat' sütununu df
DataFrame'inden çıkar ve geriye kalan sütunları X adlı değişkene
ata
y = df["fiyat"]  # 'fiyat' sütununu df DataFrame'inden al ve y
adlı değişkene ata
```

```
from sklearn.model_selection import RandomizedSearchCV  # RandomizedSearchCV'yi içe aktar, hiperparametre araması için kullanılır
from sklearn.preprocessing import MinMaxScaler  # MinMaxScaler'ı içe aktar, verileri ölçeklendirmek için kullanılır
from sklearn.model_selection import train_test_split  # train_test_split'i içe aktar, veriyi eğitim ve test setlerine bölmek için kullanılır
import xgboost as xgb  # xgboost kütüphanesini içe aktar
from sklearn.metrics import r2_score  # r2_score metriğini içe aktar, modelin performansını değerlendirmek için kullanılır

target = 'fiyat'  # Hedef değişkeni tanımla
y = df[target].values  # Hedef değişkeni y'ye ata
X = df.drop([target], axis=1)  # Hedef değişken hariç tüm değişkenleri X'e ata

# Veriyi ölçeklendirme
scaler = MinMaxScaler()  # MinMaxScaler nesnesini oluştur
X = scaler.fit_transform(X)  # X verilerini MinMaxScaler ile ölçeklendir

# Eğitim ve test setlerine bölme
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=144, shuffle=True)  # Veriyi %75 eğitim ve %25 test setlerine böl

# XGBoost için hiperparametreler
params = {
    'colsample_bytree': [0.4, 0.5, 0.6, 0.7],  # Her ağaç için kullanılacak sütun örnekleme oranları
    'learning_rate': [0.01, 0.03, 0.05, 0.07, 0.09],  # Öğrenme oranları
    'max_depth': [3, 4, 5, 6, 7],  # Ağaçların maksimum derinlikleri
    'n_estimators': [500, 1000, 1500, 2000, 2500],  # Ağaç sayıları
    'subsample': [0.6, 0.7, 0.8, 0.9, 1.0],  # Örnekleme oranları
    'gamma': [0, 0.1, 0.2, 0.3]  # Ağaç yapılandırmasında düğüm bölünmesinin minimum kayıp azaltımı
}
xgb_model = xgb.XGBRegressor(early_stopping_rounds=10)  # XGBRegressor modelini tanımla ve erken durdurma turunu ayarla

# RandomizedSearchCV ile en iyi parametreleri bulma
random_search = RandomizedSearchCV(
    xgb_model, param_distributions=params, n_iter=50, cv=5, scoring='r2', verbose=1, n_jobs=-1, random_state=42
```

```
)  # RandomizedSearchCV'yi tanımla, n_iter=50 ile 50 farklı
parametre kombinasyonunu dene
random_search.fit(x_train, y_train, eval_set=[(x_test, y_test)],
verbose=False)  # Modeli eğitim seti ile eğit ve test seti ile
doğrula

# En iyi parametreleri ve en iyi skoru yazdırma
print(f"Best params for XGBRegressor:
{random_search.best_params_}")  # En iyi parametreleri yazdır
print(f"Best R2 score: {random_search.best_score_}")  # En iyi R2
skorunu yazdır

# En iyi parametrelerle eğitilen modeli kullanarak tahmin yapma
best_xgb_model = random_search.best_estimator_  # En iyi modeli
al
y_pred = best_xgb_model.predict(x_test)  # Test seti üzerinde
tahmin yap
print(f"XGBRegressor R2_Score ---> {r2_score(y_test,
y_pred)}")  # Tahminlerin R2 skorunu yazdır
```

```
print(df.dtypes)
```

```
# Hedef değişken ve özellikleri ayırma
target = 'fiyat'  # Hedef değişkeni belirt
y = df[target].values  # Hedef değişkeni y'ye ata
X = df.drop([target], axis=1)  # Hedef değişkeni dışındaki tüm
özellikleri X'e ata

# Veriyi ölçeklendirme
scaler = MinMaxScaler()  # MinMaxScaler nesnesini oluştur
X = scaler.fit_transform(X)  # Veriyi MinMaxScaler ile
ölçeklendir

# Eğitim ve test setlerine bölme
x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=144, shuffle=True)  # Veriyi eğitim
ve test setlerine bölmek için train_test_split kullan

# XGBoost için belirlenen hiperparametreler
params = {
    'subsample': 0.8,  # Örnekleme oranı
    'n_estimators': 2500,  # Ağaç sayısı
    'max_depth': 7,  # Ağaçların maksimum derinliği
    'learning_rate': 0.03,  # Öğrenme oranı
    'gamma': 0,  # Ağaç yapılandırmasında düğüm bölünmesinin
minimum kayıp azaltımı
```

```
    'colsample_bytree': 0.5  # Her ağaç için kullanılacak sütun örnekleme oranı
}

# XGBoost modeli
best_xgb_model = xgb.XGBRegressor(**params)  # XGBoost Regressor modelini belirtilen hiperparametrelerle tanımla

# Modeli eğitme
best_xgb_model.fit(x_train, y_train)  # Modeli eğitim verileriyle eğit

# Modelin performansını değerlendirme
y_pred = best_xgb_model.predict(x_test)  # Test seti üzerinde tahmin yap
r2 = r2_score(y_test, y_pred)  # R2 skorunu hesapla
print(f"XGBRegressor R2_Score ---> {r2}")  # R2 skorunu yazdır
```