```
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 XGBRegresso
```

```
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
```

```python
def compML(df, target, alg, params=None):
    # Veriyi hedef değişken ve özellikler olmak üzere ayır
    y = df[target].values
    X = df.drop([target], axis=1)

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

    # Eğitim ve test setlerine ayır
    x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=144, shuffle=True)

    if params:
        # Eğer hiperparametreler belirtilmişse, GridSearchCV
kullanarak en iyi parametreleri bul
        grid_search = GridSearchCV(alg(), param_grid=params,
cv=5, scoring='r2')
        grid_search.fit(x_train, y_train)
        model = grid_search.best_estimator_  # En iyi modeli seç
        print(f"Best params for {alg.__name__}:
{grid_search.best_params_}")  # En iyi parametreleri yazdır
    else:
        # Hiperparametreler belirtilmemişse, algoritmayı
varsayılan parametrelerle eğit
        model = alg().fit(x_train, y_train)

    # Modeli kullanarak tahmin yap
    y_pred = model.predict(x_test)
    r2 = r2_score(y_test, y_pred)  # R2 skorunu hesapla
    print(alg.__name__, "R2_Score ---> ", r2)  # Modelin adı ve
R2 skorunu yazdır

# Modellerin listesini güncelle ve örnek hiperparametreler
tanımla
models = [LinearRegression, DecisionTreeRegressor,
KNeighborsRegressor, MLPRegressor, RandomForestRegressor,
GradientBoostingRegressor, SVR, XGBRegressor]
xgb_params = {
    "colsample_bytree":[0.4,0.5,0.6],
    "learning_rate":[0.01,0.02,0.09],
    "max_depth":[2,3,4,5,6],
    "n_estimators":[100,200,500,2000]
}

# Her model için compML fonksiyonunu çağırarak performansı
karşılaştır
for i in models:
    if i == XGBRegressor:
        compML(df, "fiyat", i, params=xgb_params)  # XGBRegressor için özel hiperparametrelerle çağır
    else:
        compML(df, "fiyat", i)  # Diğer modeller için varsayılan hiperparametrelerle çağır
```