



MAKİNE ÖĞRENMESİ OPERASYONLARI PROJE

Semih Gül

Veri Seti Hakkında

Bu Global Müşteri Terk Veriseti, çeşitli endüstrilerde müşteri terk davranışını anlamak ve tahmin etmek amacıyla titizlikle oluşturulmuştur. Demografi, ürün etkileşimleri ve bankacılık davranışları dahil olmak üzere ayrıntılı müşteri profilleriyle, bu veri seti, risk altındaki müşterileri belirlemeye ve hedefe yönelik müşteri bağlılığı stratejileri geliştirmeye yönelik makine öğrenimi modelleri geliştirmek için değerli bir kaynaktır.

Veri Açıklaması:

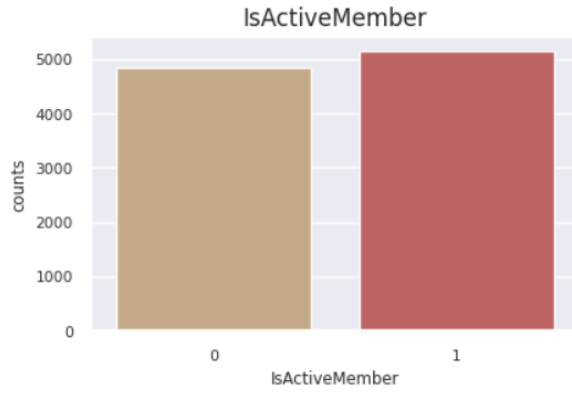
Veri kümesinin her bir sütununun neyi temsil ettiğini ayrıntılı olarak açıklaması:

- **RowNumber:** Veri kümesindeki her bir satır için benzersiz bir tanımlayıcı.
- **CustomerId:** Benzersiz müşteri kimlik numarası.
- **Surname:** Müşterinin soyadı (gizlilik nedenleriyle, eğer henüz yapılmamışsa bu veriyi anonim hale getirmeyi düşünün).
- **CreditScore:** Veri toplama zamanındaki müşterinin kredi puanı.
- **Geography:** Müşterinin ülkesi veya bölgesi, terk ile ilgili konum bazlı eğilimler hakkında içgörüler sağlar.
- **Gender:** Müşterinin cinsiyeti.
- **Age:** Müşterinin yaşı, demografik analiz için değerlidir.
- **Tenure:** Müşterinin bankayla geçirdiği yıl sayısı.
- **Balance:** Müşterinin hesap bakiyesi.
- **NumOfProducts:** Müşterinin satın aldığı veya abone olduğu ürün sayısı.
- **HasCrCard:** Müşterinin kredi kartı olup olmadığını gösterir (1 ise var, 0 ise yok).
- **IsActiveMember:** Müşterinin aktif bir üye olup olmadığını gösterir (1 ise evet, 0 ise hayır).
- **EstimatedSalary:** Müşterinin tahmini maaşı.
- **Exited:** Hedef değişken, müşterinin terkettiğini veya etmediğini gösterir.

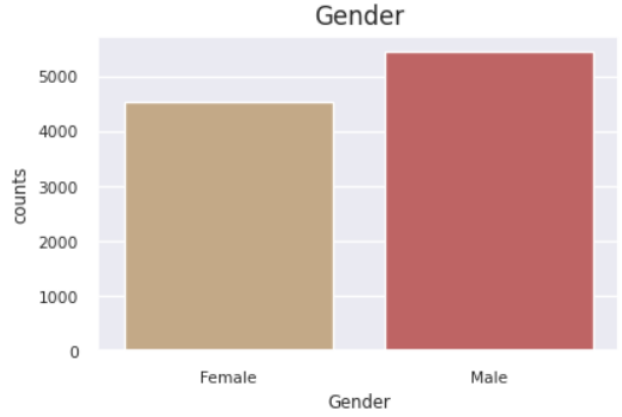
Bu veri kümesi, keşifsel veri analizi, müşteri segmentasyonu, terk davranışını tahmin etmeye yönelik modelleme ve müşteri bağlılığı stratejileri geliştirme için uygundur. İş stratejistleri, veri bilimciler ve müşteri sadakatini artırmak ve terk oranlarını azaltmakla ilgilenen araştırmacılar için zengin içgörüler sunar.

Bu rapor, müşteri kaybı (churn) tahminine yönelik bir makine öğrenimi modelinin geliştirilmesi, eğitim süreci ve dağıtımını içeren MLOps ve MLflow tabanlı bir projenin detaylarını ele almaktadır. Bu projede veri ön işleme, model eğitimi, model değerlendirme, hiperparametre optimizasyonu ve modelin bir API üzerinden dağıtılması adımları gerçekleştirilmiştir.

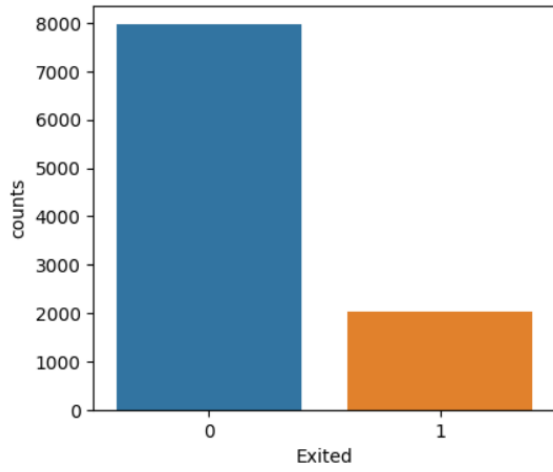
Veri Setiyle İlgili İstatistikler



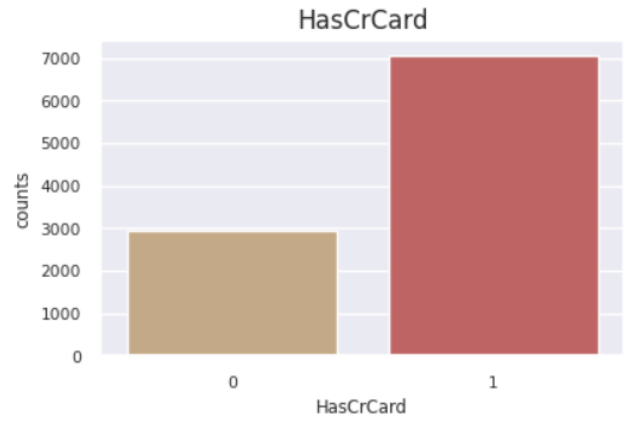
Aktif müşteri olup olmama dağılımı



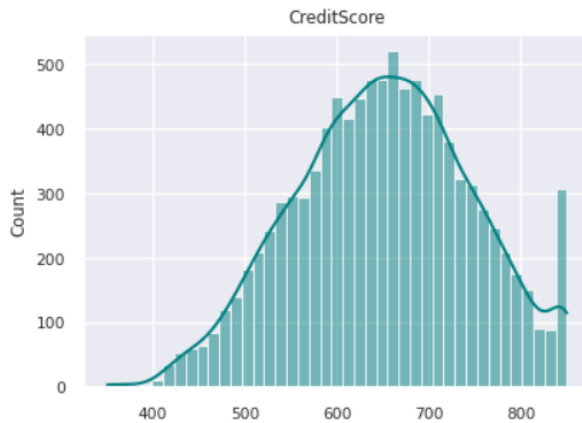
Müşterilerin cinsiyet dağılımı



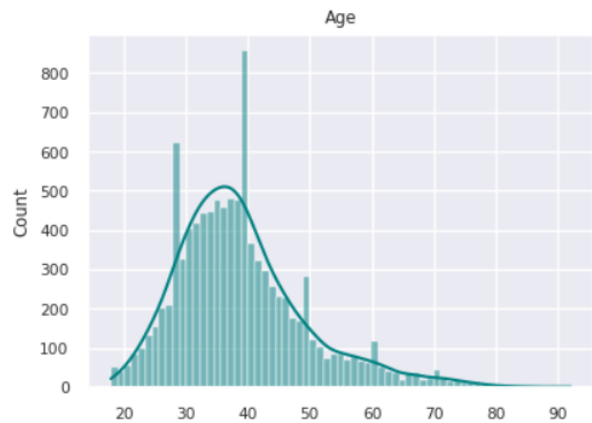
Terk eden ve etmeyen müşteri dağılımı

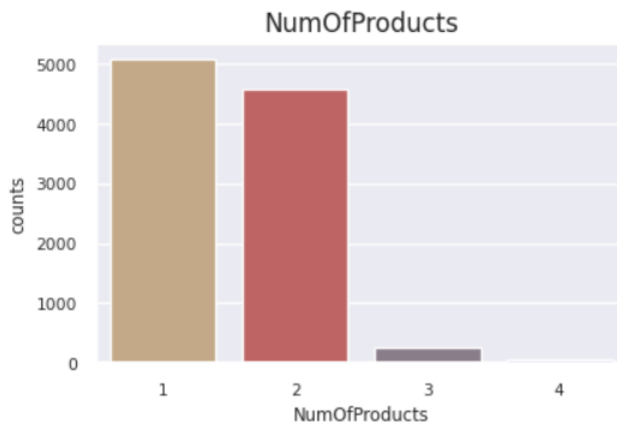


Müşterilerin kredi kartına sahip olup olmama durumları

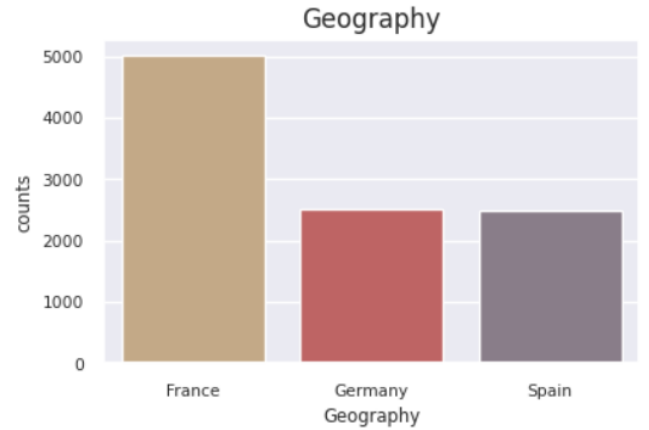


Kredi skorlarının ve yaşların dağılımı

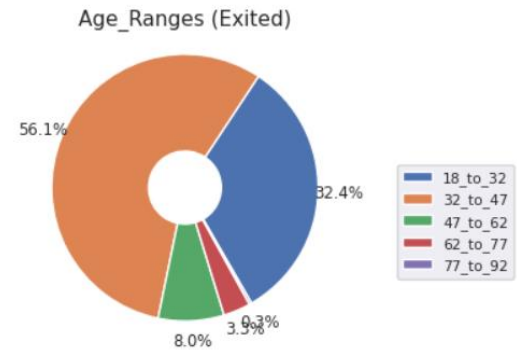
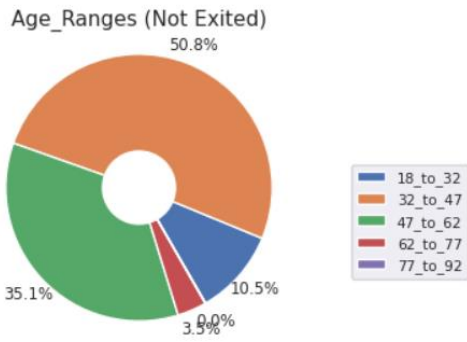




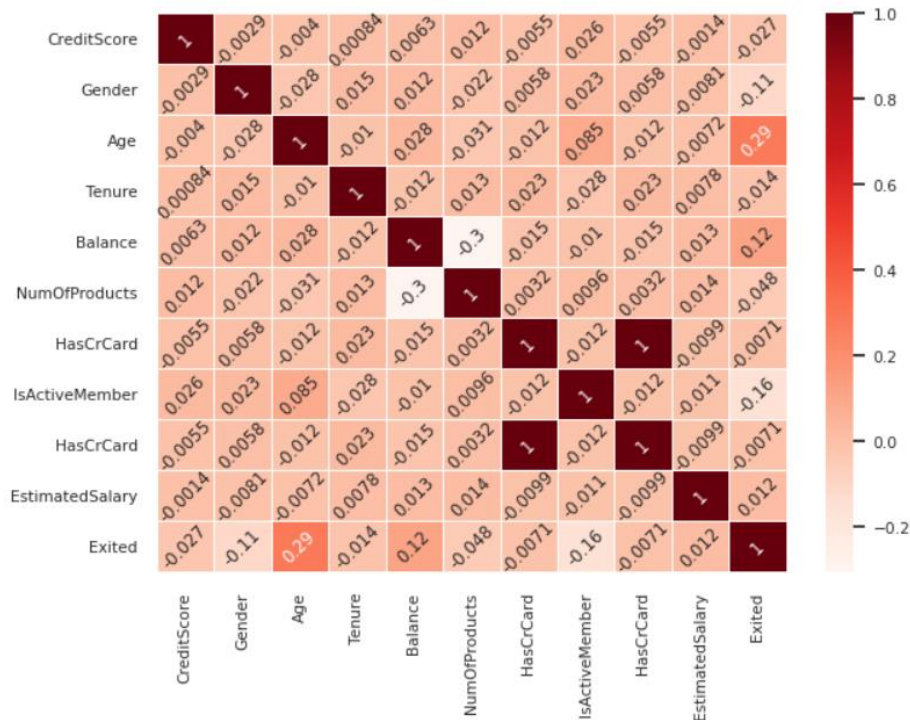
Müşterilerin sahip olduğu ürün (Abonelik) sayıları



Müşteri ülkeleri dağılımı



Yaşa göre müşterilerin terk eden müşterilerin veya terk etmeyen müşterilerin dağılımları



Veri setindeki özelliklerin korelasyon matrisi

1. Veri Ön İşleme (data_preprocessing.py)

Bu adımda, veri seti üzerinde çeşitli ön işleme teknikleri uygulanarak model eğitimi için uygun hale getirilmiştir.

a. Veri Alma ve Hazırlama

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from imblearn.over_sampling import SMOTE

# Veri Alma
1 usage  1 semihgul5 *
def load_data(file_path):
    return pd.read_csv(file_path)
```

Bu fonksiyon, CSV dosyasından veriyi yükler. Veri setinin yolunu parametre olarak alır ve pandas DataFrame olarak döner.

b. Veri Ön İşleme

```
def preprocess_data(df):
    # Sütunların ayrıştırılması
    X = df.drop(['RowNumber', 'Geography', 'CustomerId', 'Surname', 'Exited'], axis=1)
    y = df['Exited']

    # Kategorik verilerin kodlanması
    label_encoder = LabelEncoder()
    X['Gender'] = label_encoder.fit_transform(X['Gender'])

    # Verilerin ölçeklendirilmesi
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # SMOTE uygulaması
    smote = SMOTE(random_state=42)
    X_resampled, y_resampled = smote.fit_resample(X_scaled, y)

    return train_test_split(X_resampled, y_resampled, test_size=0.15, random_state=42, stratify=y_resampled)
```

- **Sütunların Ayrıştırılması:** Model eğitiminde kullanılmayacak sütunlar (RowNumber, Geography, CustomerId, Surname, Exited) çıkarılır. Exited sütunu ise hedef değişken (etiket) olarak ayrılır.
- **Kategorik Verilerin Kodlanması:** Gender sütunu, LabelEncoder kullanılarak sayısal değerlere dönüştürülür.
- **Verilerin Ölçeklendirilmesi:** StandardScaler kullanılarak veriler standart ölçeklendirilir.
- **SMOTE Uygulaması:** Dengesiz sınıfları dengelemek için SMOTE (Synthetic Minority Over-sampling Technique) kullanılır.
- **Veri Ayırma:** Eğitim ve test setlerine ayrılır.

c. Veriyi Kaydetme

```
if __name__ == "__main__":
    df = load_data('C:/Users/ABREBO/Desktop/Churn_Modelling.csv')
    X_train, X_test, y_train, y_test = preprocess_data(df)
    # Eğitim ve test verilerini kaydetme
    pd.DataFrame(X_train).to_csv('X_train.csv', index=False)
    pd.DataFrame(X_test).to_csv('X_test.csv', index=False)
    pd.DataFrame(y_train).to_csv('y_train.csv', index=False)
    pd.DataFrame(y_test).to_csv('y_test.csv', index=False)
```

Veri işlendikten sonra, eğitim ve test setleri CSV dosyalarına kaydedilir.

2. Model Eğitimi ve Hiperparametre Optimizasyonu (train_model.py)

Farklı algoritmalarla eğitilmiş birden fazla modelin birleştirilmesi, daha güçlü bir tahminleme modeli oluşturmamıza yardımcı olabilir. Örneğin, XGBoost, Random Forest algoritmalar kullanarak daha güçlü bir tahminleme modeli oluşturulabilir. Bu adımda, XGBoost modeli kullanılarak model eğitimi ve hiperparametre optimizasyonu gerçekleştirilmiştir.

a. Verileri Yükleme

```
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import mlflow.sklearn

# Verileri Yükleme
4 usages  👤 semihgul5
def load_data(file_path):
    return pd.read_csv(file_path)
```

Bu fonksiyon, eğitim ve test verilerini CSV dosyalarından yükler.

b. Hiperparametre Optimizasyonu

Modelin performansını artırmak için hiperparametre ayarlaması yapabiliriz. Örneğin, derin öğrenme modellerinde öğrenme oranı, batch boyutu ve düzenleme parametreleri gibi hiperparametreleri ayarlayarak, modelin daha iyi performans göstermesini sağlayabiliriz.

Modelin ana parametresi değildir ancak modelin performansını önemli derecede etkiler. Hyperparameter değiştirilerek en iyi performans aranır. Brute force, Random Search, Grid Search, Bayes Optimizasyon algoritmalarından biri seçilecek ve kullanılacaktır.

```
if __name__ == "__main__":
    X_train = load_data('X_train.csv')
    X_test = load_data('X_test.csv')
    y_train = load_data('y_train.csv').values.ravel()
    y_test = load_data('y_test.csv').values.ravel()

    mlflow.set_tracking_uri("http://localhost:5000")
    mlflow.set_experiment("Default")

    # Grid Search ile en iyi parametreleri bulma
    param_grid = {
        'max_depth': [3, 4, 5],
        'learning_rate': [0.1, 0.01, 0.001],
        'n_estimators': [100, 200, 300]
    }
    grid_search = GridSearchCV(xgb.XGBClassifier(), param_grid, cv=5)
    grid_search.fit(X_train, y_train)

    best_params = grid_search.best_params_
```

- **Grid Search:** GridSearchCV kullanılarak, XGBoost modelinin en iyi hiperparametreleri belirlenir. Parametre ızgarası (param_grid) farklı max_depth, learning_rate ve n_estimators değerlerinden oluşur.

c. Model Eğitimi ve Değerlendirme

```
with mlflow.start_run():  
    # En iyi parametreleri kaydetme  
    for param_name, param_value in best_params.items():  
        mlflow.log_param(param_name, param_value)  
  
    # En iyi parametrelerle modeli eğitme  
    model = xgb.XGBClassifier(**best_params)  
    model.fit(X_train, y_train)  
  
    # Tahminler  
    y_pred = model.predict(X_test)  
  
    # Değerlendirme Metrikleri  
    accuracy = accuracy_score(y_test, y_pred)  
    precision = precision_score(y_test, y_pred)  
    recall = recall_score(y_test, y_pred)  
    f1 = f1_score(y_test, y_pred)  
  
    # Metrikleri Kaydetme  
    mlflow.log_metric(key="accuracy", accuracy)  
    mlflow.log_metric(key="precision", precision)  
    mlflow.log_metric(key="recall", recall)  
    mlflow.log_metric(key="f1_score", f1)  
  
    # Karar matrisi oluşturma  
    confusion = confusion_matrix(y_test, y_pred)  
    tn, fp, fn, tp = confusion.ravel()  
  
    # Karar matrisini MLflow'a kaydetme  
    mlflow.log_metric(key="True Negatives", tn)  
    mlflow.log_metric(key="False Positives", fp)  
    mlflow.log_metric(key="False Negatives", fn)  
    mlflow.log_metric(key="True Positives", tp)  
  
    # Modeli Kaydetme ve İsimlendirme  
    model_name = "XGBoost_model"  
    mlflow.sklearn.log_model(model, model_name)
```


- **MLflow ile İzleme:** mlflow.start_run() ile yeni bir koşu başlatılır. En iyi hiperparametreler ve modelin değerlendirme metrikleri MLflow'a kaydedilir.
- **Model Eğitimi:** En iyi parametrelerle model eğitilir ve test verisi üzerinde tahminler yapılır.
- **Değerlendirme:** accuracy, precision, recall, ve f1_score metrikleri hesaplanır ve MLflow'a kaydedilir.
- **Karar Matrisi:** Karar matrisi (confusion_matrix) oluşturulur ve her bir elemanı (True Negatives, False Positives, False Negatives, True Positives) MLflow'a kaydedilir.
- **Model Kaydı:** Eğitilen model MLflow aracılığıyla kaydedilir.

3. Model Dağıtımı (app.py)

Bu adımda, Flask kullanılarak bir API oluşturulmuş ve modelin dağıtımı gerçekleştirilmiştir.

Bu Flask uygulaması, bir XGBoost modelini kullanarak müşteri kaybı tahminleri yapar. Uygulama, JSON formatında gelen veriler üzerinde tahmin işlemi gerçekleştirir ve performans metriklerini MLflow kullanarak izler.

Kullanılan Teknolojiler ve Kütüphaneler

- **Flask:** Web tabanlı uygulamalar geliştirmek için kullanılan mikro framework.
- **Pickle:** Python nesnelerini serileştirme ve tersine serileştirme için kullanılan modül.
- **NumPy:** Matematiksel işlemler için kullanılan kütüphane.
- **Scikit-learn:** Makine öğrenmesi modelleri ve veri işleme araçları sağlar.
- **Pandas:** Veri analizi ve manipülasyonu için kullanılan kütüphane.
- **MLflow:** Makine öğrenmesi yaşam döngüsünü yönetmek için kullanılan platform.

```
from flask import Flask, request, jsonify
import pickle
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import pandas as pd
import mlflow
import os

app = Flask(__name__)

# Model yükleme
model_path = os.path.join('C:/Users/ABREBO/PycharmProjects/mlops_project/mlruns/0/98225331a9a5408bae10eddc016fd2f4/artifacts/XGBoost_model/model.pkl')
model = pickle.load(open(model_path, 'rb'))
scaler = StandardScaler()

# Koşu ID'si (Run ID) belirleme
RUN_ID = "0d940e2c39654fcda0e62b54eb0a5869"

# Gerçek değerler ve tahminler için listeler
true_values = []
predictions = []
```

- '/' Route: Ana sayfa, bir hoş geldiniz mesajı döner.
- '/predict' Route: POST isteği ile gelen verileri işler, model ile tahmin yapar ve performans metriklerini günceller.
- Model ve ölçekleyici (scaler) dosya sisteminden yüklenir. pickle modülü, model dosyasını yüklemek için kullanılır.
- MLflow üzerinde metriklerin izlenmesi için kullanılacak koşu ID'si belirlenir.

- Performans metriklerinin hesaplanması için gerçek değerler ve tahminler tutulur.
- Uygulama, localhost üzerinde 5001 portunda başlatılır.

```
# Flask rotaları
# semihgul5
@app.route('/')
def home():
    return "Hello, this is the XGBoost prediction service!"

# semihgul5
@app.route(rule="/predict", methods=['POST'])
def predict():
    global true_values, predictions
    data = request.json
    features = ['CreditScore', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']

    try:
        # Veriyi uygun formata getirme
        input_data = [[data[feature] for feature in features]]
        df_input = pd.DataFrame(input_data, columns=features)

        # Kategorik verilerin kodlanması
        df_input['Gender'] = LabelEncoder().fit_transform(df_input['Gender'])

        # Verilerin ölçeklendirilmesi
        input_scaled = scaler.fit_transform(df_input)

        # Tahmin yap
        prediction = model.predict(input_scaled)
        result = prediction[0]

        # Gerçek değerleri elde etme
        if 'true_value' in data:
            true_value = data['true_value']
```

```
def predict():
    true_values.append(true_value)
    predictions.append(result)

    # Performans metriklerini hesapla
    accuracy = accuracy_score(true_values, predictions)
    precision = precision_score(true_values, predictions, zero_division=0)
    recall = recall_score(true_values, predictions, zero_division=0)
    f1 = f1_score(true_values, predictions, zero_division=0)

    # Mevcut koşuyu başlat ve metrikleri güncelle
    with mlflow.start_run(run_id=RUN_ID):
        mlflow.log_metric(key="accuracy", accuracy)
        mlflow.log_metric(key="precision", precision)
        mlflow.log_metric(key="recall", recall)
        mlflow.log_metric(key="f1_score", f1)
    else:
        # 'true_value' sağlanmamışsa bir uyarı döndür
        return jsonify({'error': 'true_value is missing from the request'}), 400

    return jsonify({'prediction': int(result)})

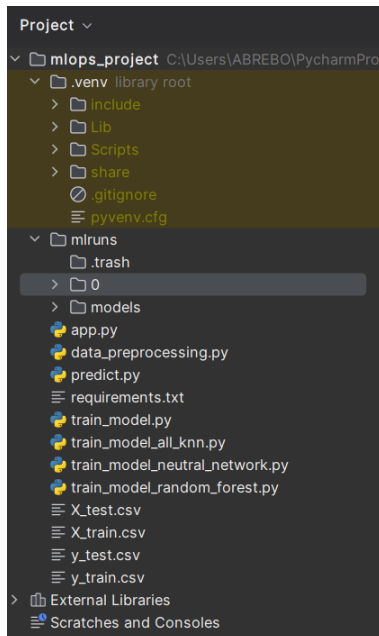
except KeyError as e:
    return jsonify({'error': f'Missing key in the request data: {e}'}), 400

except Exception as e:
    return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(port=5001, debug=True)
```

- **Veri Alma ve Hazırlama:** POST isteği ile gelen JSON verileri alınır ve işlenir. Gerekli özellikler seçilir ve uygun formata getirilir. LabelEncoder ile cinsiyet (Gender) kategorik verisi sayısal formata dönüştürülür. Veriler StandardScaler ile ölçeklendirilir.
- **Tahmin Yapma:** İşlenmiş veriler model kullanılarak tahmin edilir.
- **Performans Metriklerinin Güncellenmesi:** Gerçek değerler (true_values) ve tahminler (predictions) kullanılarak performans metrikleri hesaplanır ve MLflow'a kaydedilir. accuracy_score, precision_score, recall_score, f1_score metrikleri hesaplanır ve MLflow'a loglanır.
- **Hata Yönetimi:** Girdi verilerinde eksiklik veya hata olduğunda uygun hata mesajları döndürülür.
- **Flask Uygulaması:** Flask kullanılarak bir web servisi oluşturulur. Bu servis, /predict endpoint'ine gelen POST istekleriyle tahminler yapar.
- **Model Yükleme:** Eğitilmiş model, belirtilen yol üzerinden yüklenir.
- **Tahmin:** Gelen veriler ölçeklendirilir ve model kullanılarak tahmin yapılır.
- **Gerçek Değerler ve Tahminler:** Eğer istek verisinde gerçek değer (true_value) sağlanmışsa, gerçek değer ve tahminler listelere eklenir. Bu listeler kullanılarak performans metrikleri hesaplanır ve MLflow'a kaydedilir. Bu sayede modelin performansı mlflow üzerinden izlenebilir hale getirilir.
- **Performans Metrikleri:** accuracy, precision, recall, ve f1_score metrikleri hesaplanır ve mevcut koşuya eklenir.

Bu proje, MLOps uygulamalarının nasıl gerçekleştirilebileceğini ve MLflow ile model izleme ve kayıt işlemlerinin nasıl yapılacağını göstermektedir. Veri ön işleme, model eğitimi, hiperparametre optimizasyonu ve model dağıtım adımlarını içeren kapsamlı bir MLOps süreci oluşturulmuştur. Bu süreç, makine öğrenimi modellerinin üretim ortamında güvenilir ve izlenebilir bir şekilde kullanılmasını sağlar.



Yanda mlops projenin dosya yapısı görünmektedir.

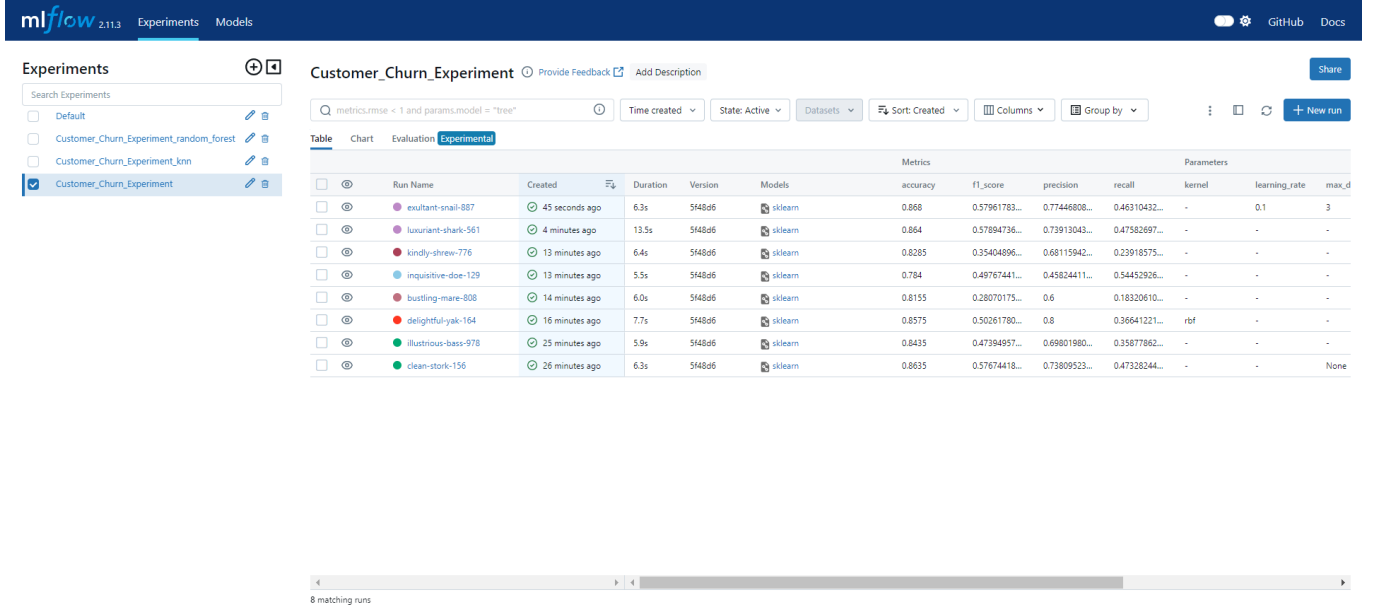
1. **.venv:** Python sanal ortamının (virtual environment) kurulduğu yerdir. Sanal ortam, bir proje için izole bir Python çalışma ortamı sağlar. Bu, proje bağımlılıklarının yalnızca belirli bir proje için yüklendiği ve başka projeleri etkilemediği anlamına gelir. .venv klasörü genellikle proje klasöründe bulunur ve projenin Python bağımlılıklarını içeren bir site-packages alt dizinine sahiptir.

2. **MLruns:** MLflow'un çalışma zamanında tutulan bir veri deposudur. MLflow, çalışma zamanında model eğitimi, parametre optimizasyonu ve metrik izleme gibi işlemleri izlemek ve yönetmek için kullanılır. Bu işlemlerin sonuçları ve kayıtları mlruns klasörü altında saklanır.

Bu klasör genellikle proje dizininin içinde bulunur ve MLflow tarafından oluşturulan çalışma zamanı verilerini içerir. Bu veriler, MLflow tarafından yapılan deneylerin sonuçları, model kayıtları, metrikler, parametreler ve diğer izleme bilgilerini içerebilir.

Bu yapı, MLflow'un deneylerin, modellerin ve işlemlerin izlenmesi, karşılaştırılması ve yönetilmesi için kullanılan bir merkezi veri deposunu sağlar. Bu sayede, ML projelerinin daha organize bir şekilde yönetilmesi ve sonuçlarının daha kolay analiz edilmesi sağlanır

3. **.py:** Modeli geliştirmek için yapılan preprocess, train, app, test ve train csv dosyalarını içerir.



The screenshot shows the MLflow Experiments interface. On the left, there's a sidebar with 'Experiments' and 'Models' tabs. The 'Experiments' tab is active, showing a list of experiments. The 'Customer_Churn_Experiment' is selected. The main area displays a table of runs for this experiment. The table has columns for Run Name, Created, Duration, Version, Models, Metrics, and Parameters. The 'Metrics' column is expanded, showing accuracy, f1_score, precision, and recall. The 'Parameters' column is also expanded, showing kernel, learning_rate, and max_d.

Run Name	Created	Duration	Version	Models	Metrics	Parameters
exultant-snail-887	45 seconds ago	6.3s	5f48d6	sklearn	accuracy: 0.868, f1_score: 0.57961783..., precision: 0.77446808..., recall: 0.46310492...	kernel: -, learning_rate: 0.1, max_d: 3
luxuriant-shark-561	4 minutes ago	13.5s	5f48d6	sklearn	accuracy: 0.864, f1_score: 0.57894736..., precision: 0.73913043..., recall: 0.47582697...	kernel: -, learning_rate: -, max_d: -
kindly-shrew-776	13 minutes ago	6.4s	5f48d6	sklearn	accuracy: 0.8285, f1_score: 0.35404896..., precision: 0.68115942..., recall: 0.23918575...	kernel: -, learning_rate: -, max_d: -
inquisitive-doe-129	13 minutes ago	5.5s	5f48d6	sklearn	accuracy: 0.784, f1_score: 0.49767441..., precision: 0.45824411..., recall: 0.54452926...	kernel: -, learning_rate: -, max_d: -
bustling-mare-808	14 minutes ago	6.0s	5f48d6	sklearn	accuracy: 0.8155, f1_score: 0.28070175..., precision: 0.6, recall: 0.18320610...	kernel: -, learning_rate: -, max_d: -
delightful-yak-164	16 minutes ago	7.7s	5f48d6	sklearn	accuracy: 0.8575, f1_score: 0.50261780..., precision: 0.8, recall: 0.36641221...	kernel: rbf, learning_rate: -, max_d: -
illustrious-bass-978	25 minutes ago	5.9s	5f48d6	sklearn	accuracy: 0.8435, f1_score: 0.47394957..., precision: 0.69801980..., recall: 0.35877862...	kernel: -, learning_rate: -, max_d: -
clean-stork-156	26 minutes ago	6.3s	5f48d6	sklearn	accuracy: 0.8635, f1_score: 0.57674418..., precision: 0.73809523..., recall: 0.47328244...	kernel: -, learning_rate: -, max_d: None

Yukarıda, geliştirilen modellerin mlflow üzerinden takibi gözükmektedir. Gözüken modellerin her biri farklı modeli – algoritmayı temsil etmektedir. Bu sayede mlflow bize en iyi modeli seçmek ve kullanmak için bir arayüz sunar.

Kullanılan modeller şunlardır;

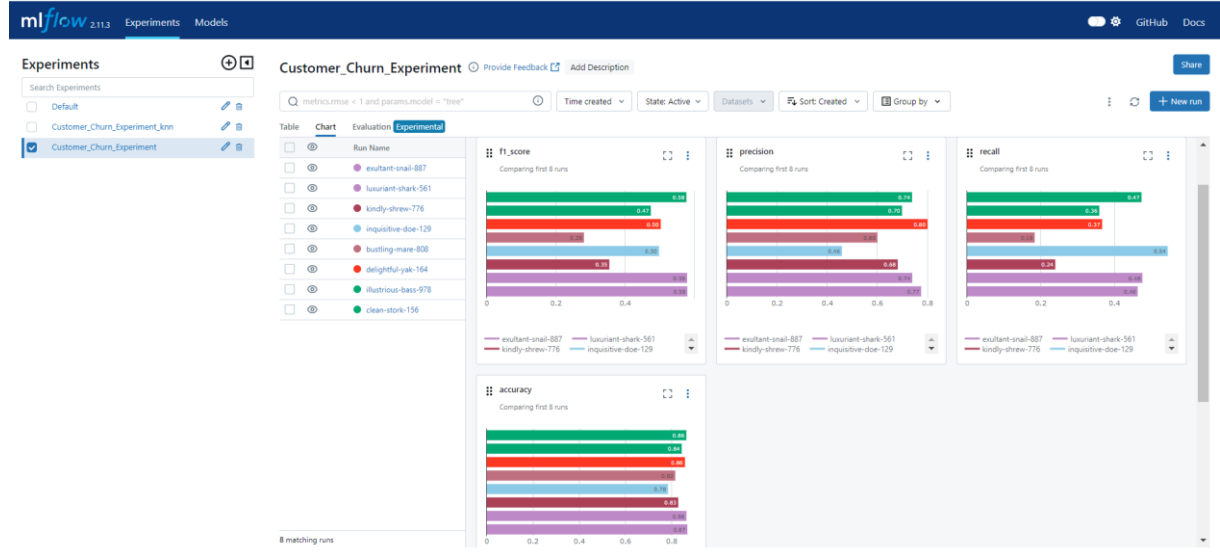
- XGBoost
- K-Nearest Neighbors (KNN)
- Decision Trees
- Random Forest
- Support Vector Machines (SVM)
- Logistic Regression
- Naive Bayes
- Neural Networks

Bu modeller arasından mlflow sayesinde en iyi model seçimi yapılacaktır. Bunun için ise, duration, accuracy, f1 skor, precision ve recall skorları değerlendirmeye alınacaktır.

Duration: MLflow tarafından bir işlemin başlangıç ve bitiş arasındaki süreyi ölçmek için kullanılan bir ölçüdür. Özellikle bir ML modelinin eğitim süresini veya bir deneyin çalışma süresini ölçmek için kullanılır.

Bir MLflow çalışması başladığında, bir zaman damgası (timestamp) alınır. Çalışma tamamlandığında, başlangıç zamanından itibaren geçen toplam süre hesaplanır ve bu süre "duration" olarak kaydedilir.

Bu, MLflow aracılığıyla gerçekleştirilen deneylerin ve işlemlerin süresinin izlenmesini sağlar ve bu sürelerin performans değerlendirilmesi ve kaynak planlaması için kullanılmasına olanak tanır.

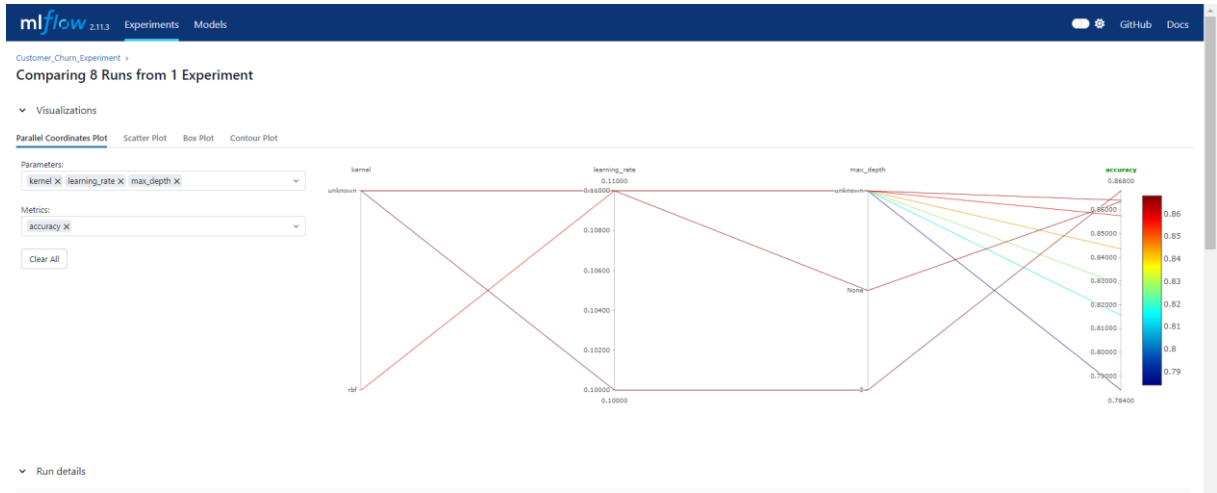


Modellerin skorlarının grafiksel olarak karşılaştırılması.

MLflow, çeşitli grafikler ve görselleştirmeler oluşturmak için kullanılabilen bir grafik arayüzü sunar. MLflow'un grafik özellikleri, deneylerin sonuçlarını, model performansını, parametrelerin dağılımını ve daha fazlasını görselleştirmek için kullanılabilir.

MLflow'un grafik yetenekleri arasında şunlar bulunur:

- Deney İzleme Grafikleri: MLflow, deneylerin sonuçlarını izlemek için farklı grafik türlerini destekler. Bunlar arasında metrik eğrileri, parametrelerin dağılımları, model performansı ve daha fazlası bulunur.
- Model Görselleştirme: MLflow, eğitilmiş modelleri görselleştirmek için grafikler sağlar. Bu, modelin mimarisini, özellik ağırlıklarını, özellik önem sıralamasını ve diğer özellikleri görselleştirmeyi içerebilir.
- Hipotez Testi Görselleştirme: MLflow, farklı model sürümleri arasında karşılaştırmalar yapmak ve sonuçları görselleştirmek için hipotez testi grafiklerini sağlar.
- Parametre Optimizasyonu Görselleştirme: MLflow, hiperparametre optimizasyon sonuçlarını ve modellerin performansını görselleştirmek için grafikler sunar.
- MLflow'un grafik özellikleri, deneylerin sonuçlarını anlamak, model performansını izlemek ve eksiklikleri belirlemek için kullanılabilir. Bu görselleştirmeler, MLflow'un eksikliklerini belirlemek ve model eğitim sürecini iyileştirmek için önemli bir araçtır.



Paralel koordinat grafiği, çoklu değişkenler arasındaki ilişkileri görselleştirmek için kullanılan bir grafik türüdür. Bu grafikte, her değişken bir çizgi olarak temsil edilir ve bu çizgiler birbirine paralel olarak yerleştirilir. Her bir veri noktası, bu çizgiler arasında bir yol oluşturur. Bu yol, veri noktasının her bir değişken için aldığı değeri gösterir.

Paralel koordinat grafiği, özellikle çok boyutlu veri setlerini analiz etmek ve farklı değişkenler arasındaki ilişkileri keşfetmek için kullanışlıdır. Özellikle her bir veri noktasının diğer değişkenlerle nasıl ilişkilendiğini görselleştirmek için etkilidir. Bununla birlikte, çok sayıda değişken içeren veri setlerinde paralel koordinat grafiği okunması zor olabilir ve aşırı karmaşık hale gelebilir. Bu durumda, daha az değişken içeren alt küme veri setleriyle çalışmak daha etkili olabilir.

Smote tekniği ile modelin performansının artırılması

Default

Provide Feedback

Add Description

Share

Q metrics.rmse < 1 and params.model = "tree"

Time created

State: Active

Datasets

Sort: Created

Columns

Group by

+ New run

Table

Chart

Evaluation

Experimental

	Run Name	Created	dataset	Duration	Source	Models	Metrics			
							accuracy	f1_score	precision	recall
	puzzled-lynx-0	15 seconds ago		8.9s	C:\Users...	sklearn	0.88279614...	0.87889273...	0.90876565...	0.85092127...
	classy-fowl-365	5 days ago		8.5s	C:\Users...	sklearn	0.85266666...	0.54620123...	0.73480662...	0.43464052...
	Production Model Evaluat...	5 days ago		16.3min	C:\Users...	-	0.75	0	0	0
	stately-moose-187	6 days ago		6.1s	C:\Users...	xgb_resp_ai_model v1	0.865	0.56451612...	0.77092511...	0.44529262...
	suave-snake-712	6 days ago		7.8s	C:\Users...	xgb v1	0.866	0.57961783...	0.77446808...	0.46310432...
	abundant-perch-739	6 days ago		7.0s	C:\Users...	sklearn	0.864	0.57366771...	0.74693877...	0.46564885...
	likeable-trout-145	6 days ago		7.3s	C:\Users...	sklearn	0.864	0.57366771...	0.74693877...	0.46564885...
	bedecked-ant-712	6 days ago		6.8s	C:\Users...	sklearn	0.864	0.57366771...	0.74693877...	0.46564885...

Veri setimizde müşterinin terk ettiği ve terk etmediği (0,1) sınıfları arasında büyük fark olduğunu gözlemledikten sonra smote kullanılmasına karar verildi.

SMOTE (Sentetik Azınlık Aşırı Örnekleme Tekniği): Sınıf dengesizliği sorununu çözmek için kullanılan bir tekniktir. Sınıf dengesizliği, bir sınıfın diğerinden önemli ölçüde daha az veriye sahip olduğu durumları ifade eder. SMOTE, özellikle makine öğrenimi modellerinin eğitildiği sınıflandırma problemlerinde sıkça karşılaşılan bir sorunu ele alır.

SMOTE, azınlık sınıfındaki örneklerin sayısını artırmak için sentetik örnekler oluşturur. Bu, azınlık sınıfındaki örneklerle benzer ancak hafifçe değiştirilmiş sentetik örnekler üreterek yapılır. SMOTE, azınlık sınıfındaki örnekler arasında rastgele bir şekilde örnekleme yapar ve her örnek için bu





örnekler arasında rasgele bir komşu seçer. Seçilen örnekler arasındaki farkları dikkate alarak yeni sentetik örnekler oluşturur.

Bu teknik kullanıldıktan sonra tüm metrik skorlarında büyük oranda artış gözlemlendi. Örneğin, F1 skor 0,55 değerinden 0,88 'e çıktı. Bu da tekniğin ne kadar önemli bir oranda başarılı olduğunu ve kullanımının önemini göstermektedir.

Model performansının api sorguları ile mlflow üzerinden izlenmesi

Projemize uygun bir serving tasarımı olarak "Stateless Serving Function" yöntemini seçelim. Bu yöntem, her kullanıcı isteği için model tahminlerini sağlar ve herhangi bir kullanıcıya özgü durum veya geçmiş bilgi gerektirmez. Her bir istekte bağımsız bir tahmin yapılır.

Bu yöntemi uygulamak için, Flask kullanarak bir API oluşturabiliriz. Flask, basit ve hafif bir web uygulama çerçevesidir ve hızlı bir şekilde servisler oluşturabilmemizi sağlar.

Table Chart Evaluation Experimental											
							Metrics				
<input type="checkbox"/>	Run Name	Created		Dataset	Duration	Source	Models	accuracy	f1_score	precision	recall
<input type="checkbox"/>	 Production Model Evaluat...	 16 minutes ago		-	163min	 C:\Users...	-	0.75	0	0	0

Geliştirilen model için bir api yazıldı ve bu api ile modele prediction atılması sağlandı. Bunun takibi için ise mlflow kullanıldı. Postman programı aracılığı ile modele bir 'post' isteği gönderilmekte, bu istek sonucunda model çalıştırılmakta ve json formatında tahminler gösterilmektedir.

Monitoring yapılabilmesi için modele her sorgu atıldığında, expected ve result değerleri işlendi. Bu, veri setinden örnekler alınarak yapıldı. Her sorgu sonucunda mlflow'da modelin tüm metrik değerleri güncellenmektedir. Örneğin yukarıdaki görselde modelimize birkaç test prediction'u attığımızda accuracy skor'u güncellenmektedir. Bu durum da data drift gibi model bozulmalarının takibi yapılması için önemli bir araç sunuyor.

POST

http://127.0.0.1:5001/predict

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1 {
2   "CreditScore": 600,
3   "Gender": "Female",
4   "Age": 42,
5   "Tenure": 2,
6   "Balance": 0,
7   "NumOfProducts": 1,
8   "HasCrCard": 1,
9   "IsActiveMember": 0,
10  "EstimatedSalary": 8.88,
11  "true_value": 0
12 }
13
```






BodyCookiesHeaders (5)Test Results

Status: 200 OKTime: 7.49 sSize: 187 BSave as example

PrettyRawPreviewVisualizeJSON

1 {
2 "prediction": 0
3 }

Reframing ile regresyon problemine çevirme

Table Chart Evaluation Experimental									
							Metrics		
<input type="checkbox"/>	Run Name	Created 	Dataset	Duration	Source	Models	mae	mse	r2
<input type="checkbox"/>	 bold-quail-248	 15 seconds ago	-	8.9s	 CIUsers...	 sklearn	0.21541586...	0.09153958...	0.63384163...

Reframing tasarım kalıbı kullanılarak sınıflandırma problemimizi bir tahmin (regresyon) problemine çevirebiliriz. Sonuç olarak r2 skoru baz aldığımızda 0,63 skorunu elde etmiş olduk.