



İSTANBUL MEDENİYET ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

FİNANS TEKNOLOJİLERİ VE YAPAY ZEKA DERSİ FINAL PROJE ÖDEVI

Abdullah BUZKAN	21120205067
Gizem Yağmur ERGELEN	21120205063
Hasret BAĞ	21120205053
Zeynep USLUBAŞ	21120606052

DERS DANIŞMANI
Zeynep AYDEMİR

KONU
Hane Halkı Gelir Analizi

Aralık, 2024
İstanbul Medeniyet Üniversitesi, İstanbul

İÇİNDEKİLER

GİRİŞ	4
1. Problem Tanıma	6
2. Veri Toplama.....	6
3. Veri Ön İşleme	6
3.1 Veri Setinin Genel Yapısı	7
3.2 Tekrar Eden Satırların Kontrolü ve Kaldırılması.....	8
3.3 NULL Değerlerin Kontrolü	8
3.4 “?” Olan Sembollerin Analizi.....	9
3.5 Kategorik Değişkenlerin Analizi	11
3.6 Nümerik Değişkenlerin Analizi.....	18
3.7 Akyarı Değer Analizi	23
3.8 Veri Ölçekleme	28
3.8.1 Normalizasyon	28
3.9 Veri Dönüşümü	29
3.9.1 Label Encoding Dönüşümü	29
3.9.2 Özellik İndirgeme (Feature Reduction)	32
4.Keşifsel Veri Analizi (EDA)	33
4.1 Yaş Dağılımı Analizi	33
4.1.2 Yaş Gruplarına Göre Gelir Dağılımı Analizi	34
4.2 Çalışma Sınıfı Dağılımı Analizi	36
4.2.1 Çalışma Sınıflarına Göre Gelir Dağılımı Analizi	37
4.3 Eğitim Seviyesi Dağılımı Analizi	39
4.3.1 Eğitim Seviyelerine Göre Gelir Dağılımı Analizi	40
4.4 İlişki Durumu Dağılımı Analizi	41
4.4.1 İlişki Durumlarına Göre Gelir Dağılımı Analizi	43
4.5 Meslek Dağılımı Analizi	44
4.5.1 Mesleklerde Göre Gelir Dağılımı Analizi	45
4.6 İrk Dağılımı Analizi	47
4.6.1 Irklara Göre Gelir Dağılımı Analizi.....	48
4.7 Cinsiyet Dağılımı Analizi	50
4.7.1 Cinsiyete Göre Gelir Dağılımı Analizi	51
4.8 Memlekete Göre Dağılım Analizi	52
4.8.1 Memlekete Göre Gelir Dağılımı Analizi	53
4.9 Normalleştirilmiş Haftalık Çalışma Saatleri Dağılımı Analizi	55
4.9.1 Haftalık Çalışma Saatleri ve Gelir Dağılımı Analizi (Violin Plot)	56

4.10 Gelir Dağılımı Analizi (Pasta Grafigi).....	58
4.11 Korelasyon Matrisi Analizi.....	59
4.12 Değişim Katsayısı (Coefficient of Variation) Analizi	60
4.13 Çarpıklık (Skewness) ve Basıklık (Kurtosis) Analizi	62
5.Veri Modelleme	64
 5.1 Algoritma Seçimi	64
 5.2 Model Eğitimi.....	65
6.Model Değerlendirme	67
 6.1 Overfitting ve Underfitting Kontrolü	68
7.Sonuçların Yorumlanması ve Görselleştirilmesi.....	69
KAYNAKÇA	72

GİRİŞ

Günümüzde, bireylerin gelir seviyelerinin tahmin edilmesi, sosyo-ekonomik araştırmaların yanı sıra, hükümetlerin ve kuruluşların politika geliştirme süreçlerinde de önemli bir rol oynamaktadır. Bu projede, bireylerin yıllık gelirlerinin belirli bir eşik olan 50.000 birimin üzerinde veya altında olup olmadığını öngörmeyi hedefliyoruz. Bu tahmin süreci, makine öğrenimi tekniklerinden biri olan ve genellikle sınıflandırma ve regresyon problemlerinde yüksek performans gösteren XGBoost algoritmasını kullanarak gerçekleştirilmektedir.

Projemiz, demografik özellikler (yaş, cinsiyet, eğitim durumu gibi) ve sosyo-ekonomik veriler (meslek, çalışma saatleri, aile durumu gibi) içeren kapsamlı bir veri setini temel almaktır ve bu veriler Kaggle platformu üzerinden sağlanan, anonimleştirilmiş yetişkin bireylerin gelir bilgilerini içeren bir veri setidir. Bu veri seti, bireylerin gelir seviyeleri üzerinde önemli bir etkiye sahip olan çeşitli faktörleri içermekte olup, modelimizin eğitimi için zengin bir kaynak sunmaktadır.

Proje kapsamında, ilk olarak veri ön işleme aşaması gerçekleştirilmiştir. Bu aşamada, eksik veya hatalı veriler temizlenmiş, kategorik veriler sayısal hale getirilmiş ve veri seti model eğitimi için uygun hale getirilmiştir. İkinci olarak, özellik mühendisliği teknikleri kullanılarak modelin başarısını artıracak yeni değişkenler türetilmiş ve mevcut özelliklerin üzerinde düzenlemeler yapılmıştır. Modelin eğitimi sırasında, çeşitli parametre ayarlamaları test edilmiş ve en iyi sonucu veren parametreler belirlenmiştir.

Sonuç olarak, bu çalışma, bireylerin gelir seviyelerinin belirlenmesinde demografik ve sosyo-ekonomik verilerin nasıl etkili kullanılabileceğini göstermektedir. Ayrıca, elde edilen sonuçlar doğrultusunda, modelin doğruluk, hassasiyet, duyarlılık ve F1 skoru gibi performans metrikleri ile değerlendirilmesi yapılmış ve modelin uygulanabilirliği test edilmiştir. Projemiz, bu tür verilerle çalışan araştırmacılar, politika yapıcılar ve ilgili diğer tüm bireyler için faydalı bilgiler ve yöntemler sunmaktadır.

XGBoost (eXtreme Gradient Boosting)

XGBoost, yüksek performanslı bir gradient boosting kütüphanesidir ve kendi içinde bir dizi ileri düzey özellik barındırır. Gelişmiş bir makine öğrenmesi algoritması olan XGBoost, özellikle sınıflandırma ve regresyon problemlerinde geniş çapta kullanılmaktadır. Gradient boosting, zayıf tahmin modellerini (genellikle karar ağaçları) sıralı bir şekilde birleştirerek güçlü bir tahmin modeli oluşturmayı amaçlar. XGBoost, bu yaklaşımı optimize eder ve ölçeklenebilirlik, taşınabilirlik ve yüksek performans sunar.

XGBoost'un Öne Çıkan Özellikleri:

- Hız ve Performans:** XGBoost, paralel işleme, ağaç budama, donanım optimizasyonu ve verimli bellek kullanımı sayesinde oldukça hızlıdır. Bu özellikler, büyük veri setlerinin hızla işlenmesini ve modelin etkili bir şekilde eğitilmesini sağlar.
- Çapraz Doğrulama Desteği:** XGBoost, modelin doğruluğunu artırmak için dahili çapraz doğrulama mekanizması sunar. Bu, her eğitim iterasyonunda modelin performansını doğrulamak ve aşırı uyumu önlemek için kullanılır.
- Esneklik:** Çeşitli özelleştirme seçenekleri sunarak farklı bilimsel ve endüstriyel problemlere uyarlanabilir. Hem sayısal hem de kategorik verilerle çalışabilir.

- Kayıp Değerlerle Başa Çıkabilme:** XGBoost, eksik verileri otomatik olarak yönetebilir ve eksik değerlerin bulunduğu verilerle etkili bir şekilde çalışabilir. Bu, veri ön işleme sürecinde ek adımların gereksiz hale gelmesini sağlar.
- Ölçeklenebilirlik:** Büyük veri setleri ve bilgisayar kaynakları üzerinde ölçeklenmek üzere tasarlanmıştır. GPU ve çoklu işlemci desteği ile donatılmıştır.
- Düzenlileştirme (Regularization):** Aşırı uyumu önlemek için L1 (Lasso Regresyonu) ve L2 (Ridge Regresyonu) düzenlileştirmelerini içerir. Bu, modelin daha genelleştirilmiş ve stabil olmasını sağlar.
- Özellik Önem Derecesi:** Hangi değişkenlerin hedef değişken üzerinde en fazla etkiye sahip olduğunu belirleme yeteneği sunar, böylece modelin yorumlanabilirliği artar.

XGBoost'un Kullanım Alanları:

XGBoost, finansal sektörden sağlık bilimlerine, perakende ve e-ticaretten üretim ve lojistike kadar çeşitli alanlarda kullanılmaktadır. Risk yönetimi, müşteri tercih analizi, ürün önerileri sistemleri ve talep tahmini gibi geniş bir uygulama yelpazesi sunar.

LİTERATÜR TARAMASI:

Yapay zekâ ve makine öğrenimi tekniklerinin finansal teknolojilerde kullanımı, özellikle demografik ve sosyo-ekonomik verilerle bireysel gelir tahminlerinde, ekonomik araştırmaların geleceğini şekillendirmektedir. Bu alandaki mevcut çalışmalar, gelir tahmini problemlerinin çözümünde klasik istatistiksel yöntemler ile makine öğrenimi modellerinin entegrasyonunun potansiyelini vurgulamaktadır.

Örneğin, Kahyun Jo'nun çalışması (2024), demografik öngörücü değişkenler kullanarak bireylerin yıllık 50.000 dolar üzeri kazanıp kazanmadığını tahmin etmeye yönelik kapsamlı bir analiz sunar. Bu çalışma, makine öğrenimi tekniklerinin, gelir seviyelerinin doğru bir şekilde tahmin edilmesinde etkili olduğunu göstermektedir. Ayrıca, Random Forest ve Neural Networks gibi algoritmaların, gelir tahmininde özellikle başarılı olduğu belirtilmektedir (Jo, 2024).

Michael Matkowski'nin (2021) yürüttüğü bir başka çalışma ise, Machine Learning modellerinin, gelir tahminlerinde geleneksel yöntemlere göre daha iyi performans sergilediğini ortaya koymaktadır. Bu çalışma, Current Population Survey verilerini kullanarak, 2017 ve 2018 verileri ile modelleri eğitmiş ve 2019-2020 verileri üzerinde bu modelleri test etmiştir. Sonuçlar, makine öğrenimi modellerinin gelir tahminlerinde yüksek başarı sağladığını göstermiştir.

Bu literatürler, XGBoost modelinin kullanımını da desteklemekte ve bu modelin, gelir tahminlerinde başarılı sonuçlar verebileceğini öne sürmektedir. XGBoost, hem yüksek hesaplama hızı hem de model doğruluğunu artırmak için kullanılan bir dizi özelliği ile bilinir. Paralel işleme, ağaç budama, verimli bellek kullanımı ve donanım optimizasyonu gibi özelliklerini, büyük veri setleri üzerinde hızlı ve etkin bir şekilde çalışmasını sağlar.

Projemizde, XGBoost modelinin kullanılması, tahminlerin doğruluğunu artırırken, aynı zamanda modelin karmaşıklığını ve eğitim süresini optimize etme potansiyelini sunmaktadır. Bu yaklaşım, demografik ve sosyo-ekonomik değişkenlerin gelir üzerindeki etkilerini daha detaylı incelememize olanak tanıyarak, politika yapıcılar ve araştırmacılar için değerli bilgiler sunmaktadır.

HANE HALKI GELİR ANALİZİ

1. Problem Tanıma

Gelir seviyesi, bireylerin yaşam standartlarını, sosyal ve ekonomik durumlarını belirlemede önemli bir göstergedir. Bu çalışmada, bireylerin belirli demografik ve sosyo-ekonomik özelliklerine dayanarak yıllık gelirlerinin 50.000 dolardan yüksek veya düşük olduğunu tahmin etmek hedeflenmiştir.

2. Veri Toplama

Bu çalışmada kullanılan veri seti, "Kaggle" platformundan alınmıştır. Veri seti, çeşitli özelliklerden (örneğin, yaşı, cinsiyet, eğitim seviyesi, meslek, çalışma saatleri vb.) oluşmakta ve önceden toplanmış, düzenlenmiş bir biçimde sağlanmıştır. Dolayısıyla, bu çalışma kapsamında doğrudan veri toplama işlemi gerçekleştirilmemiştir.

3. Veri Ön İşleme

Aşağıda gerekli kütüphaneler eklenmiştir.

- **numpy:** Büyük, çok boyutlu diziler ve matrisler üzerinde hızlı matematiksel işlemler yapmak için kullanılır; veri analizi için temel bir yapı taşı oluşturur.
- **pandas:** Verileri organize etmek, işlemek ve analiz etmek için güçlü DataFrame ve Series veri yapıları sağlar.
- **matplotlib.pyplot:** Verilerin grafiksel olarak görselleştirilmesi için temel grafik çizim araçları sunar.
- **seaborn:** Daha estetik ve karmaşık görselleştirme araçlarıyla verilerin analizine yönelik gelişmiş grafikler oluşturmak için kullanılır (Matplotlib'i temel alır).

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

Modelin doğruluğunu artırmak ve daha iyi tahmin sonuçları elde edebilmek için veri seti üzerinde çeşitli ön işleme adımları gerçekleştirilmiştir.

Veri setinin analiz ve modelleme süreçlerine uygun hale getirilmesi için öncelikle veri seti incelenmiştir.

Bu kapsamda aşağıdaki işlemler gerçekleştirilmiştir:

Bu kod "pandas" kütüphanesi kullanılarak bir CSV dosyasını okumak ve içeriğini bir DataFrame'e yüklemek için kullanılır.

Okunan veriler df adlı bir değişkene atanır. df burada bir DataFrame nesnesidir. Bu nesne üzerinde çeşitli veri manipülasyonu ve analizi işlemleri yapacağız.

```
[ ] 1 df = pd.read_csv('data.csv')
```

3.1 Veri Setinin Genel Yapısı

Veri setinin ilk 5 satırı incelendi:

Veri setinde ? olan yerler vardır. Bu genellikle bilinmeyen veya eksik veri anlamına gelebilir. Bunla ilgili analizi ilerleyen adımlarda bakacağız.

```
1 print(df.head())
2
3 age workclass fnlwgt education educational-num marital-status \
4 25 Private 226982 11th 7 Never-married
5 38 Private 89814 HS-grad 9 Married-civ-spouse
6 28 Local-gov 336951 Assoc-acdm 12 Married-civ-spouse
7 44 Private 160323 Some-college 10 Married-civ-spouse
8 18 ? 103497 Some-college 10 Never-married
9
10 occupation relationship race gender capital-gain capital-loss \
11 Machine-op-inspct Own-child Black Male 0 0
12 Farming-fishing Husband White Male 0 0
13 Protective-serv Husband White Male 0 0
14 Machine-op-inspct Husband Black Male 7688 0
15 ? Own-child White Female 0 0
16
17 hours-per-week native-country income
18 40 United-States <=50K
19 50 United-States <=50K
20 40 United-States >50K
21 40 United-States >50K
22 30 United-States <=50K
```

Veri setinin kaç satır ve sütundanoluştuğu görüntülendi:

Verimizde 48842 adet satır ve 15 adet sütun bulunmaktadır.

```
[ ] 1 df.shape
2
3 (48842, 15)
```

Veri setimizde çalışacağımız özellikler olan sütun isimleri aşağıdaki kod ile listelenmiştir:

```
[ ] 1 df.columns
2
3 Index(['age', 'workclass', 'fnlwgt', 'education', 'educational-num',
4        'marital-status', 'occupation', 'relationship', 'race', 'gender',
5        'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
6        'income'],
7       dtype='object')
```

Veri analizi sürecinde, veriyi daha iyi anlamak için değişkenler iki ana gruba ayrılmıştır: “**nümerik (sayısal)**” ve “**kategorik değişkenler**”. Bu işlem, her iki türdeki değişkenleri ayırarak, her birinin farklı analiz yöntemleriyle ele alınabilmesine olanak tanır. Veri setindeki sütunlar nümerik ve kategorik olmak üzere iki gruba ayrılmıştır:

```
1 categorical_features = [col for col in df.columns if (df[col].dtype == "object") or (df[col].dtype == "categorical")]
2
3 numerical_features = [col for col in df.columns if (df[col].dtype != "object") and (df[col].dtype != "categorical")]
4
5 categorical_features, numerical_features
6
7 ('workclass',
8  'education',
9  'marital-status',
10 'occupation',
11 'relationship',
12 'race',
13 'gender',
14 'native-country',
15 'income'],
16 ['age',
17  'fnlwgt',
18  'educational-num',
19  'capital-gain',
20  'capital-loss',
21  'hours-per-week'])
```

Burada kullanmış olduğumuz info() komutu ile veri setinin genel bir yapısını inceledik ve hangi veri türleriyle çalıştığımızı görmüş olduk.

```
1 print(df.info())
2
3 <class 'pandas.core.frame.DataFrame'
4 RangeIndex: 48842 entries, 0 to 48841
5 Data columns (total 15 columns):
6 #   Column      Non-Null Count  Dtype  
7 ---  --          --          --    
8 0   age         48842 non-null   int64  
9 1   workclass   48842 non-null   object 
10 2  fnlwt       48842 non-null   int64  
11 3  education    48842 non-null   object 
12 4  educational-num 48842 non-null   int64  
13 5  marital-status 48842 non-null   object 
14 6  occupation   48842 non-null   object 
15 7  relationship  48842 non-null   object 
16 8  race         48842 non-null   object 
17 9  gender       48842 non-null   object 
18 10 capital-gain 48842 non-null   int64  
19 11 capital-loss 48842 non-null   int64  
20 12 hours-per-week 48842 non-null   int64  
21 13 native-country 48842 non-null   object 
22 14 income       48842 non-null   object 
23 dtypes: int64(6), object(9)
24 memory usage: 5.6+ MB
25 None
```

Yukarıda yapılmış olan tüm işlemler, veri setinin genel yapısını anlamaya, eksiklikleri tespit etmeye ve sonraki veri ön işleme adımlarını planlamaya yardımcı olması için yapılmıştır.

3.2 Tekrar Eden Satırların Kontrolü ve Kaldırılması

Veri setinde tekrarlayan satırlar tespit edilmiştir:

Üzerinde çalışacağımız veri setinde 52 adet duplicate veri bulunmaktadır. Bu veriler, modelin doğruluğu ve genel performansı olumsuz etkileyebilir. Bu nedenle veri setinden duplicate satırların temizlenmesi önemlidir.

```
[ ] 1  duplicate_rows = df.duplicated()
2  print(duplicate_rows.sum())
3
4 → 52
```

Aşağıdaki kod ile veri setindeki yalnızca duplicate olan ilk satırı bırakır ve diğerlerini veri setinden kaldırır.

```
[ ] 1  df = df.drop_duplicates()
```

Duplicate satırların başarıyla kaldırıldığından emin olmak için yeniden kontrol yapılmıştır:

Sonucun 0 olduğunu görerek temizlendiğini doğruladık.

```
1 1  duplicate_rows = df.duplicated()
2 2  print(duplicate_rows.sum())
3
4 → 0
```

3.3 NULL Değerlerin Kontrolü

NULL değerler, bir veri setinde eksik veya tanımlanmamış verileri ifade eder. Bu tür eksik veriler, analiz ve modelleme süreçlerinde sorunlara yol açabilir. Bu nedenle veri setindeki NULL değerlerin kontrolü önemli bir adımdır.

Her sütundaki null değerlerin sayısına baktık ve veri setinde NULL değer bulunmadığı tespit edilmiştir.

```
1 df.isnull().sum()
```

	0
age	0
workclass	0
fnlwgt	0
education	0
educational-num	0
marital-status	0
occupation	0
relationship	0
race	0
gender	0
capital-gain	0
capital-loss	0
hours-per-week	0
native-country	0
income	0

dtype: int64

3.4 “?” Olan Sembollerin Analizi

Veri setinde NULL değer yoktur ancak “?” olarak ifade edilen değerler vardır. Bu geçersiz değerlerin tespiti için aşağıdaki kod kullanılmıştır:

```
1 df.isin(['?']).sum()
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	True	False	False	False	False	True	False	False	False	False	False	False	False	False
...
48837	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
48838	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
48839	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
48840	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
48841	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

48790 rows x 15 columns

Veri setindeki ‘?’ simgelerinin hangi sütunlarda ve ne kadar yoğunlukta bulunduğu tespit ettiğimizde.

“?” değeri veri setindeki toplamda 3 sütunda bulunmakta.

```
1 df.isin(['?']).sum()
```

	0
age	0
workclass	2795
fnlwgt	0
education	0
educational-num	0
marital-status	0
occupation	2805
relationship	0
race	0
gender	0
capital-gain	0
capital-loss	0
hours-per-week	0
native-country	856
income	0

dtype: int64

“?” değeri içeren sütunlarda bu değerler NULL (NaN) olarak değiştirilmiştir:

Bu adımlar, **workclass**, **occupation** ve **native-country** sütunlarında “?” yerine NaN (NULL) değerlerini atar.

```
▶ 1 df['workclass']=df['workclass'].replace('?',np.nan)
  2 df['occupation']=df['occupation'].replace('?',np.nan)
  3 df['native-country']=df['native-country'].replace('?',np.nan)
```

“?” değerlerinin başarıyla NULL değerlerine dönüştürülp dönüştürülmediğini kontrol etmek için bu kodu yazdık.

Bu işlem sonucunda veri setinde “?” değerlerinin ortadan kalktığı gözlemlenmiştir.

```
▶ 1 df.isin(['?']).sum()
```

```
→      0
age      0
workclass      0
fnlwgt      0
education      0
educational-num      0
marital-status      0
occupation      0
relationship      0
race      0
gender      0
capital-gain      0
capital-loss      0
hours-per-week      0
native-country      0
income      0
```

dtype: int64

Aşağıdaki kod ile NULL değerlerin hangi sütunda ve kaç tane olduğu belirlenmiştir:

Veri setindeki eksik değerler **workclass**, **occupation** ve **native-country** sütunlarında yer almaktadır.

```
▶ 1 df.isnull().sum()
```

```
→      0
age      0
workclass      2795
fnlwgt      0
education      0
educational-num      0
marital-status      0
occupation      2805
relationship      0
race      0
gender      0
capital-gain      0
capital-loss      0
hours-per-week      0
native-country      856
income      0
```

dtype: int64

Eksik verilerin (NaN) bulunduğu satırlar, dropna() metodu ile veri setinden çıkarılmıştır:

```
1 df.dropna(how='any', inplace=True)
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K
...
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	16024	0	40	United-States	>50K

NULL değerler silindiğinden sonra kalan satır ve sütun değerleri listelenmiştir:

```
1 df.shape
```

(45175, 15)

Bu kod, pandas kütüphanesi kullanılarak oluşturulan bir DataFrame içindeki sayısal sütunlar için çeşitli istatistiksel özetleri sağlar.

```
1 df.describe()
```

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
count	45175.000000	4.517500e+04	45175.000000	45175.000000	45175.000000	45175.000000
mean	38.556170	1.897388e+05	10.119314	1102.576270	88.687593	40.942512
std	13.215349	1.056524e+05	2.551740	7510.249876	405.156611	12.007730
min	17.000000	1.349200e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.173925e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783120e+05	10.000000	0.000000	0.000000	40.000000
75%	47.000000	2.379030e+05	13.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

3.5 Kategorik Değişkenlerin Analizi

Veri setindeki kategorik sütunlar listelenir:

```
1 # 1. Kategorik değişkenleri tespit et
2 categorical_features = df.select_dtypes(include=['object', 'category']).columns
3 categorical_features
```

Index(['workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'gender', 'native-country', 'income'],
dtype='object')

Kategorik değişkenlerdeki benzersiz değerleri gözlemllemek için unique() metodu kullanılmıştır.

workclass sütunu için:

```
1 unique_work=df['workclass'].unique().tolist()
2 print(unique_work)
3
4 ['Private', 'Local-gov', 'Self-emp-not-inc', 'Federal-gov', 'State-gov', 'Self-emp-inc', 'Without-pay']
```

education sütunu için:

```
▶ 1 unique_ed=df['education'].unique().tolist()
  2 print(unique_ed)
→ ['11th', 'HS-grad', 'Assoc-acdm', 'Some-college', '10th', 'Prof-school', '7th-8th', 'Bachelors', 'Masters', '5th-6th', 'Assoc-voc', '9th', 'Doctorate', '12th', '1st-4th', 'Preschool']
```

marital-status sütunu için:

```
▶ 1 unique_marital=df['marital-status'].unique().tolist()
  2 print(unique_marital)
→ ['Never-married', 'Married-civ-spouse', 'Widowed', 'Separated', 'Divorced', 'Married-spouse-absent', 'Married-AF-spouse']
```

occupation sütunu için:

```
▶ 1 unique_oc=df['occupation'].unique().tolist()
  2 print(unique_oc)
→ ['Machine-op-insptn', 'Farming-fishing', 'Protective-serv', 'Other-service', 'Prof-specialty', 'Craft-repair', 'Adm-clerical', 'Exec-managerial', 'Tech-support', 'Sales', 'Priv-house-serv', 'Transport-moving', 'Handlers-cleaners', 'A...']
```

relationship sütunu için:

```
▶ 1 unique_re=df['relationship'].unique().tolist()
  2 print(unique_re)
→ ['Own-child', 'Husband', 'Not-in-family', 'Unmarried', 'Wife', 'Other-relative']
```

race sütunu için:

```
▶ 1 unique_ra=df['race'].unique().tolist()
  2 print(unique_ra)
→ ['Black', 'White', 'Other', 'Amer-Indian-Eskimo', 'Asian-Pac-Islander']
```

gender sütunu için:

```
▶ 1 unique_ge=df['gender'].unique().tolist()
  2 print(unique_ge)
→ ['Male', 'Female']
```

native-country sütunu için:

```
▶ 1 unique_nat=df['native-country'].unique().tolist()
  2 print(unique_nat)
→ ['United-States', 'Peru', 'Guatemala', 'Mexico', 'Dominican-Republic', 'Ireland', 'Germany', 'Philippines', 'Thailand', 'Haiti', 'El-Salvador', 'Puerto-Rico', 'Vietnam', 'South', 'Columbia', 'Japan', 'India', 'Cambodia', 'Poland', 'T...']
```

income sütunu için:

```
▶ 1 unique_in=df['income'].unique().tolist()
  2 print(unique_in)
→ ['<=50K', '>50K']
```

Her bir kategorik değişkenin, her bir değeri için gözlem sayıları **value_counts()** metodunu ile hesaplanmıştır. Bu işlem, her kategorik sütun için gözlem sayılarının ayrıntılı bir şekilde görüntülenmesini sağlar.

```
1 #Her bir kategorik değişken için toplam gözlem sayısını value_counts() ile bul.
2 for col in categorical_features:
3     print(f'{df[col].value_counts()}\n', 5*****)
```

→ workclass

workclass	count
Private	33262
Self-emp-not-inc	3795
Local-gov	3100
State-gov	1946
Self-emp-inc	1645
Federal-gov	1406
Without-pay	21

Name: count, dtype: int64

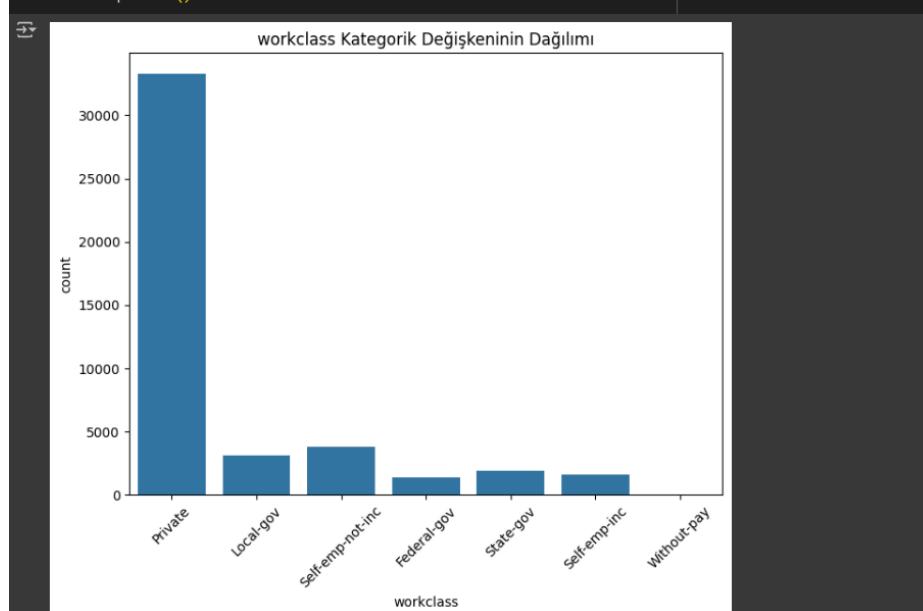
education

education	count
HS-grad	14770
Some-college	9887
Bachelors	7559
Masters	2513
Assoc-voc	1958
11th	1619
Assoc-acdm	1507
10th	1223
7th-8th	822
Prof-school	785
9th	676
12th	575
Doctorate	544
5th-6th	447
1st-4th	220
Preschool	70

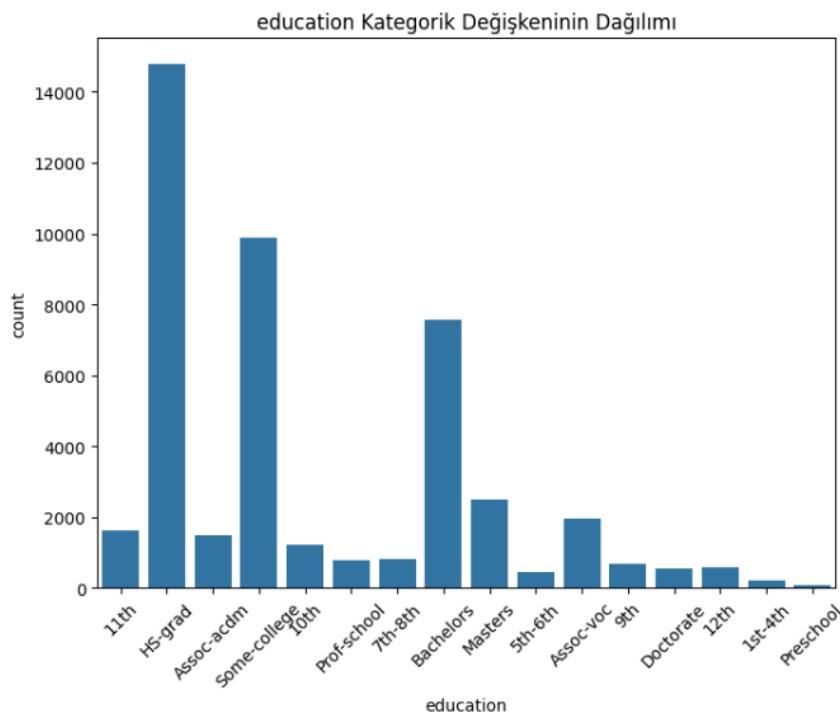
Name: count, dtype: int64

Aşağıdaki kod, kategorik değişkenlerin dağılımlarını görselleştirmek için bar grafikleri oluşturur. Her bir kategorik değişkenin değerlerinin veri setindeki sıklığını gösteren bir bar grafiği çizmektedir.

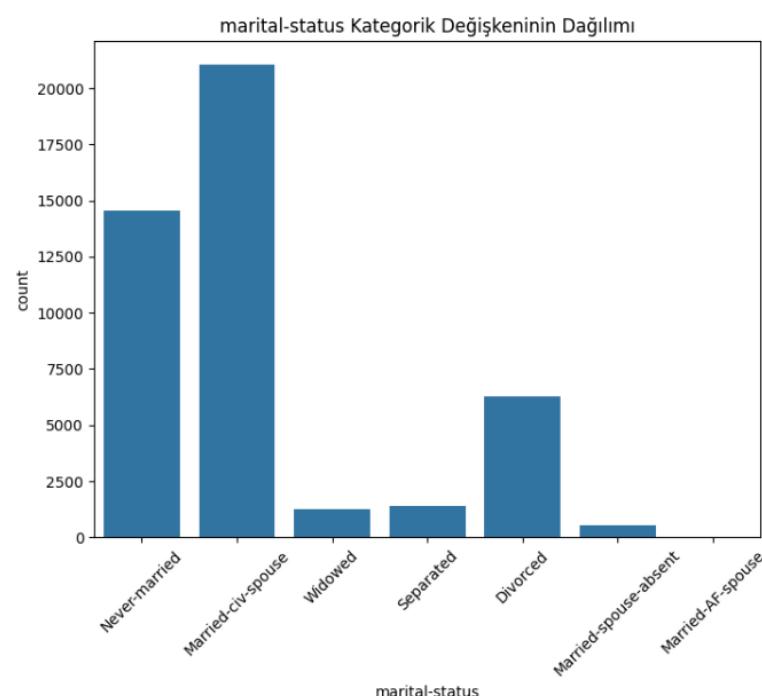
```
1 #Kategorik değişkenlerin bar grafiğini çiz
2 for col in categorical_features:
3     plt.figure(figsize=(8, 6))
4     sns.countplot(x=col, data=df)
5     plt.title(f'{col} Kategorik Değişkeninin Dağılımı')
6     plt.xticks(rotation=45) # X eksenindeki etiketlerin daha rahat okunması için
7     plt.show()
```



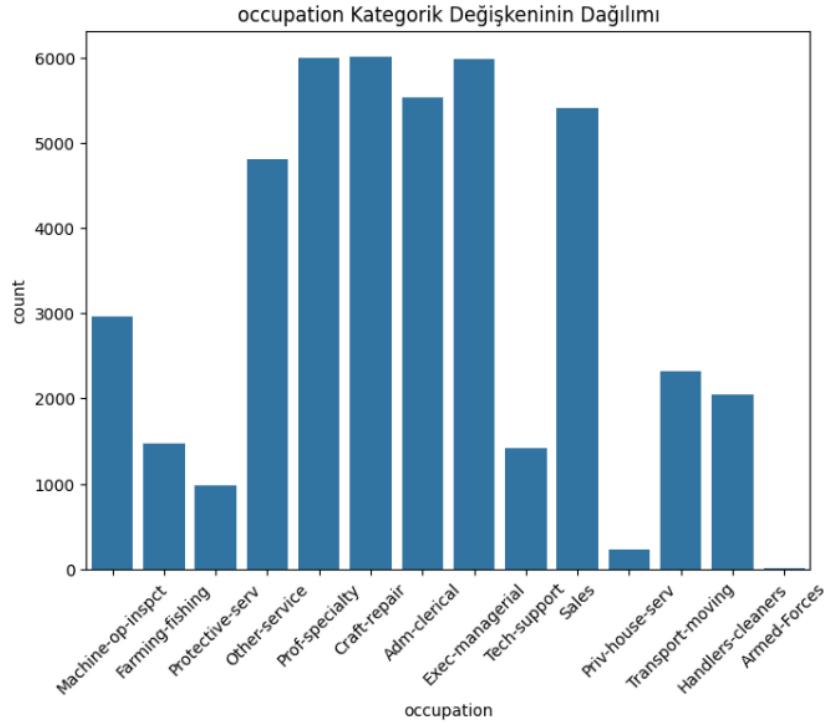
- "Private" sınıfı en baskın iş sınıfı olarak öne çıkıyor. Bu, veri kümesindeki bireylerin çoğunluğunun özel sektörde çalıştığını göstermektedir.
- Diğer sınıflar, özellikle "Without-pay" ve "Self-emp-inc" gibi sınıflar oldukça düşük oranlara sahiptir. Bu durum, bu kategorilerin veri kümesinde nadir olduğunu işaret etmektedir.



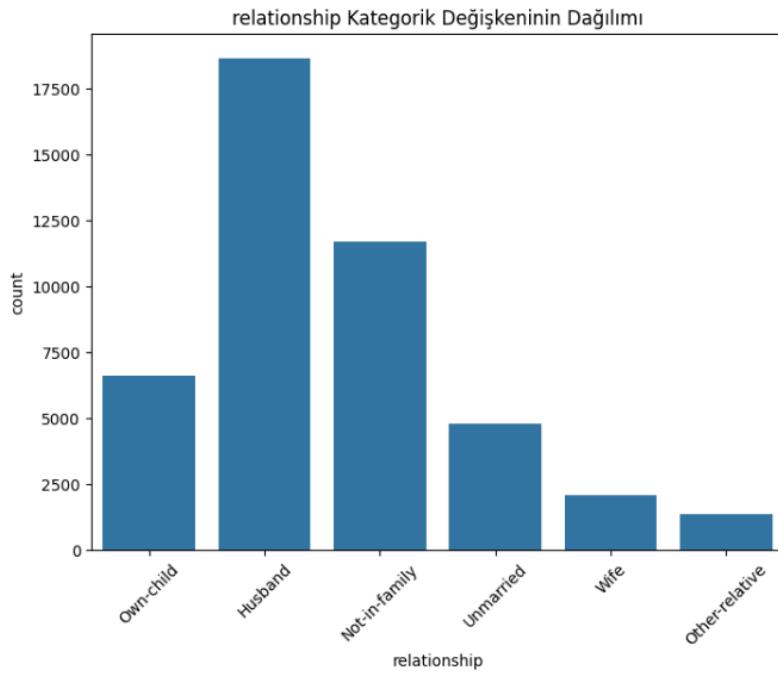
- En büyük kategori "HS-grad" (Lise mezunu), ardından "Some-college" ve "Bachelors" gelmektedir. Bu, bireylerin çoğunluğunun lise veya daha yüksek düzeyde eğitim aldığıını göstermektedir.
- "Preschool" ve "1st-4th" gibi düşük eğitim seviyeleri oldukça az temsil edilmiş. Bu, düşük eğitim seviyesine sahip bireylerin veri kümesinde nadir olduğunu işaret etmektedir.



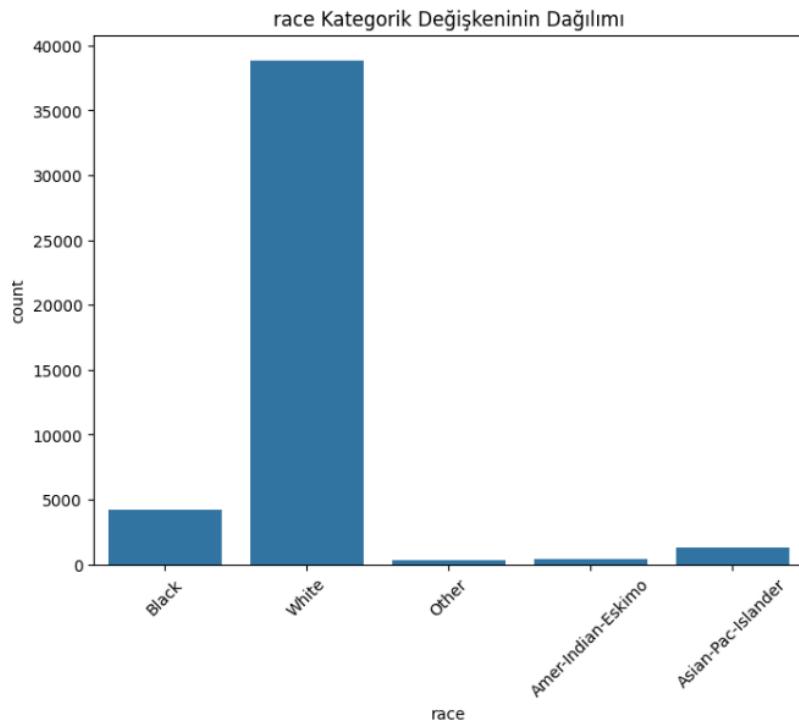
- "Married-civ-spouse" (Evli) en yaygın kategori, "Never-married" (Hiç evlenmemiş) ise ikinci sıradadır. Bu, veri kümesindeki bireylerin çoğunun evli veya hiç evlenmemiş olduğunu göstermektedir.
- "Married-spouse-absent" ve "Married-AF-spouse" gibi kategoriler oldukça nadirdir.



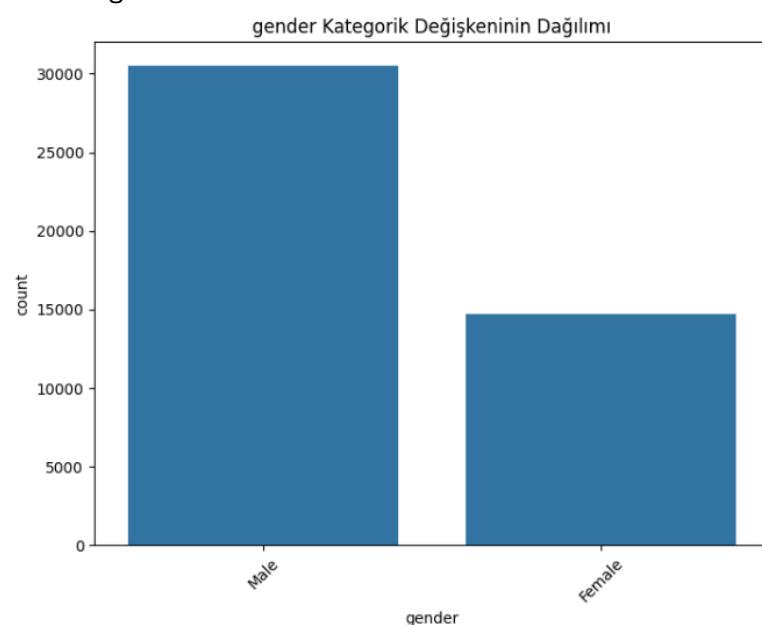
- "Prof-specialty," "Exec-managerial," ve "Craft-repair" gibi meslekler en fazla temsil edilen kategoriler arasındadır.
- "Armed-Forces" gibi meslekler ise nadirdir. Bu, veri kümesindeki bireylerin genelde profesyonel veya yönetici pozisyonlarda çalıştığını işaret etmektedir.



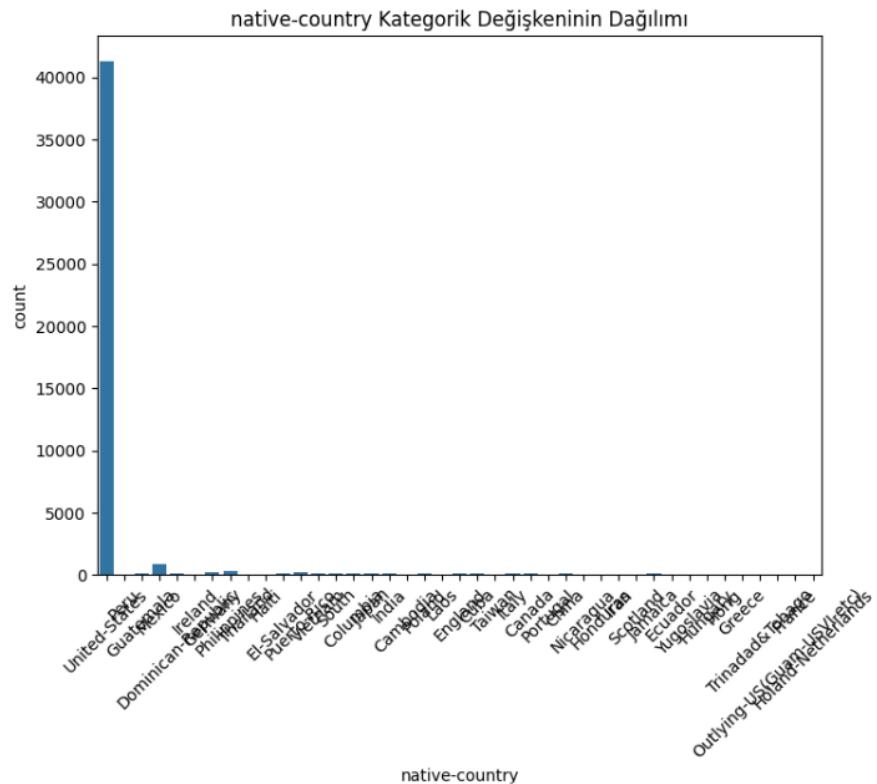
- "Husband" (Eş) kategorisi açık ara en yüksek değere sahiptir. Bu, veri kümesinde çok sayıda evli olduğunu göstermektedir.
- "Not-in-family" (Aile içinde olmayan) ikinci sırada, ardından "Unmarried" (Evlenmemiş) gelmektedir. Bu durum, bireylerin bir kısmının aile ile ilişkili olmadığını ya da yalnız yaşadığını işaret etmektedir.
- "Other-relative" ve "Wife" (Kadın eş) kategorileri düşük oranlara sahiptir.



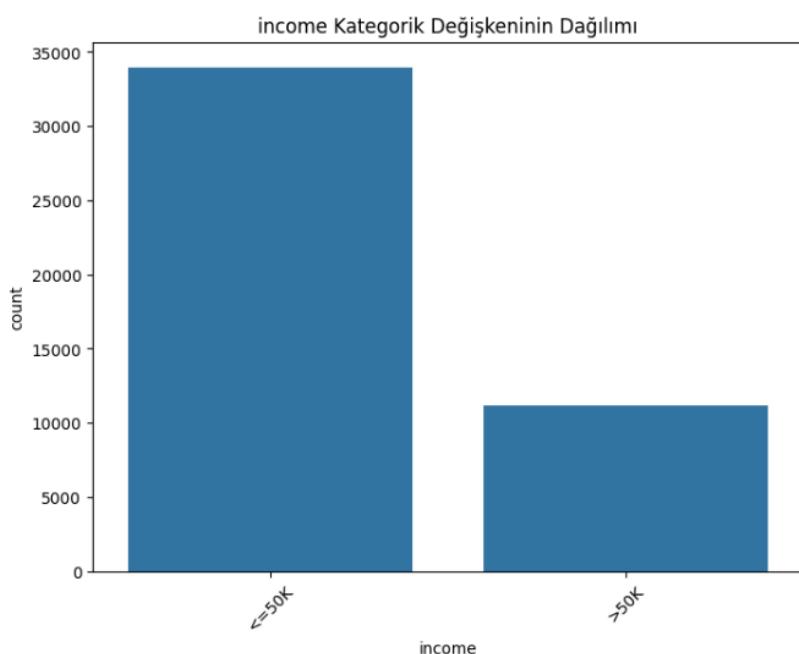
- "White" (Beyaz) kategorisi veri kümesindeki bireylerin büyük çoğunluğunu oluşturmaktadır.
- Diğer ırklar ("Black," "Asian-Pac-Islander," "Amer-Indian-Eskimo") oldukça düşük temsil edilmiştir. Bu, veri kümesinin demografik çeşitlilik açısından dengesiz olduğunu göstermektedir.



- "Male" (Erkek) kategorisi "Female" (Kadın) kategorisinden oldukça yüksektir. Veri kümesinde erkeklerin kadınlardan çok daha fazla olduğu görülmektedir.
- Bu durum, analiz yapılrken cinsiyet bazlı dengesizliği göz önünde bulundurmayı gerektirmektedir.



- "United States" (ABD) açık ara en yüksek değere sahiptir. Bu, veri kümesindeki bireylerin çoğunluğunun ABD'de doğmuş olduğunu göstermektedir.
- Diğer ülkeler oldukça düşük oranlarda temsil edilmiştir. Bu durum, veri kümesinin büyük ölçüde ABD'ye odaklandığını ve uluslararası çeşitliliğin sınırlı olduğunu işaret etmektedir.



- " $\leq 50K$ " kategorisi baskın, yani veri kümesindeki bireylerin büyük çoğunluğunun yıllık geliri 50 bin doların altındadır.
- " $>50K$ " (50 bin dolar üzeri) kategorisi çok daha düşüktür. Bu, gelir dağılımında belirgin bir eşitsizliği göstermektedir.

Bu analiz, veri kümesinin demografik ve ekonomik özellikleri hakkında önemli bilgiler sunmaktadır. İş sınıfı, eğitim, medeni durum, meslek, ilişki durumu, ırk, cinsiyet, doğum yeri ve gelir gibi değişkenlerin dağılımları incelenmiş ve veri kümesinin genel olarak ABD, beyaz, erkek bireylerden oluştuğu, özel sektörde çalışıldığı ve gelir seviyesinin çoğunlukla düşük olduğu belirlenmiştir. Bu bulgular, analiz sırasında dikkate alınması gereken önemli veri dengesizliklerini ortaya koymaktadır. Bu dengesizlikler, yapılacak analizlerde yanlılıklarını önlemek adına dikkatle değerlendirilmelidir.

3.6 Nümerik Değişkenlerin Analizi

Bu işlem sırasında float64 ve int64 veri tipindeki kolonlar filtrelenmiş ve bu kolonlardan oluşan yeni bir veri çerçevesi (df_numerical) oluşturulmuştur. İlk beş satır, genel bir ön izleme sunmak için kullanılmıştır.

```
[1] # Veri setindeki numerik kolonları seçmek için aşağıdaki yapıyı kullanabiliriz.
[2] df_numerical = df.select_dtypes(include = ["float64", "int64"])
[3] df_numerical.head()
```

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
0	25	226802	7	0	0	40
1	38	89814	9	0	0	50
2	28	336951	12	0	0	40
3	44	160323	10	7688	0	40
5	34	198693	6	0	0	30

Numerik değişkenlerin temel istatistiksel değerleri, describe() metodu ile çıkarılmıştır. Bu özet tablo, her değişken için şu bilgileri içerir: Değer sayısı(count), ortalama(mean), standart sapma(std), minimum ve maksimum değerler(min, max) ve çeyreklikler(25%, 50%, 75%).

Bu özet tablo, her bir değişkenin dağılımını ve temel özelliklerini anlamamıza olanak tanır.

```
[1] df_numerical.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	45175.0	38.556170	13.215349	17.0	28.0	37.0	47.0	90.0
fnlwgt	45175.0	189738.798450	105652.436515	13492.0	117392.5	178312.0	237903.0	1490400.0
educational-num	45175.0	10.119314	2.551740	1.0	9.0	10.0	13.0	16.0
capital-gain	45175.0	1102.576270	7510.249876	0.0	0.0	0.0	0.0	99999.0
capital-loss	45175.0	88.687593	405.156611	0.0	0.0	0.0	0.0	4356.0
hours-per-week	45175.0	40.942512	12.007730	1.0	40.0	40.0	45.0	99.0

Yaş (Age)

- Bireylerin yaş ortalaması 38,56 olarak hesaplanmıştır. Veri kümesinde en küçük yaş 17, en büyük yaş ise 90 olarak belirlenmiştir.
- Çeyrek değerlere göre, bireylerin %25'i 28 yaşın altında, %50'si (medyan) 37 yaşın altında ve %75'i 47 yaşın altındadır. Bu durum, veri kümesinin genel olarak genç ve orta yaş grubu bireylerden oluştuğunu göstermektedir.

Final Ağırlık (Fnlwgt)

- Final ağırlık değişkeninin ortalaması 189738,8, standart sapması ise 105652,43 olarak bulunmuştur. Bu durum, değişkenin geniş bir dağılıma sahip olduğunu göstermektedir.
- Minimum değer 13492, maksimum değer ise 1490400 olarak kaydedilmiştir. Bu, veri kümelerinde üç değerlerin bulunduğuunu işaret etmektedir.

Eğitim Süresi (Educational-num)

- Eğitim süresi ortalaması 10,12 olarak hesaplanmıştır. Bu, bireylerin genelde lise mezuniyetine yakın bir eğitim seviyesine sahip olduğunu göstermektedir.
- Eğitim süresi değişkeninde minimum değer 1 yıl, maksimum değer ise 16 yıl olarak belirlenmiştir. Medyan değer 10 yıl olup, bireylerin çoğunluğunun lise mezunu olduğunu göstermektedir.

Sermaye Kazancı (Capital-gain)

- Sermaye kazancının ortalaması 1102,58 olarak bulunmuş, ancak standart sapma 7510,25 ile yüksek bir varyasyon göstermektedir.
- Çoğu birey (çeyrek değerlere göre %75'i) sermaye kazancı elde etmemiştir. Maksimum değer ise 99999 olup, bu değişkenin üç değerlere sahip olduğunu ortaya koymaktadır.

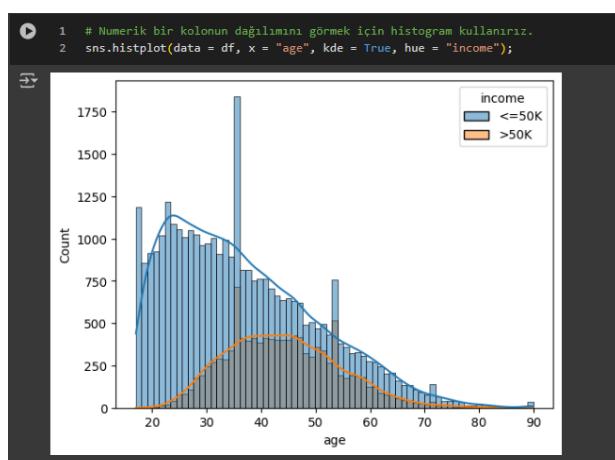
Sermaye Kaybı (Capital-loss)

- Sermaye kaybı değişkeninin ortalaması 88,69 olarak hesaplanmıştır. Çoğu birey için sermaye kaybı sıfırdır (çeyrek değerler de sıfırdır).
- Maksimum değer 4356 olup, bu değişkende de nadiren yüksek değerler görüldüğü tespit edilmiştir.

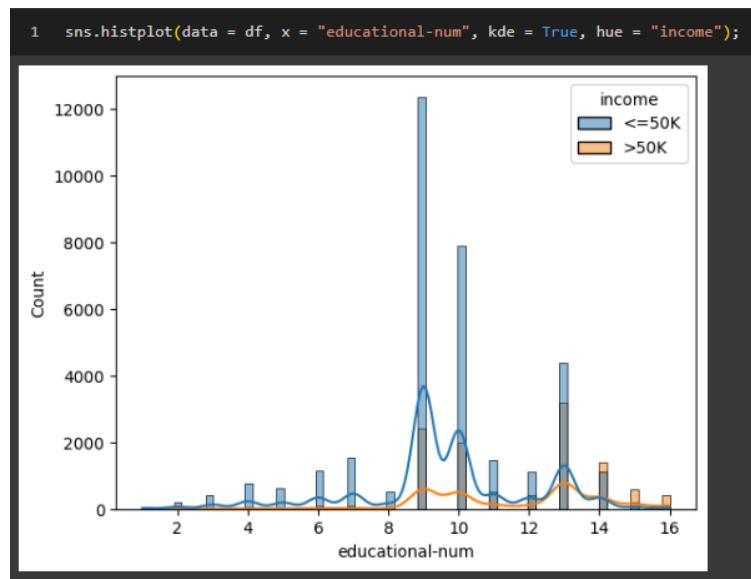
Haftalık Çalışma Süresi (Hours-per-week)

- Haftalık çalışma süresi ortalaması 40,94 olarak belirlenmiştir. Minimum çalışma süresi 1 saat, maksimum çalışma süresi ise 99 saatdir.
- Çeyrek değerlere göre, bireylerin %50'si haftada 40 saat çalışmaktadır. Bu, veri kümelerindeki bireylerin çoğunluğunun standart çalışma saatlerine uyduğunu göstermektedir.

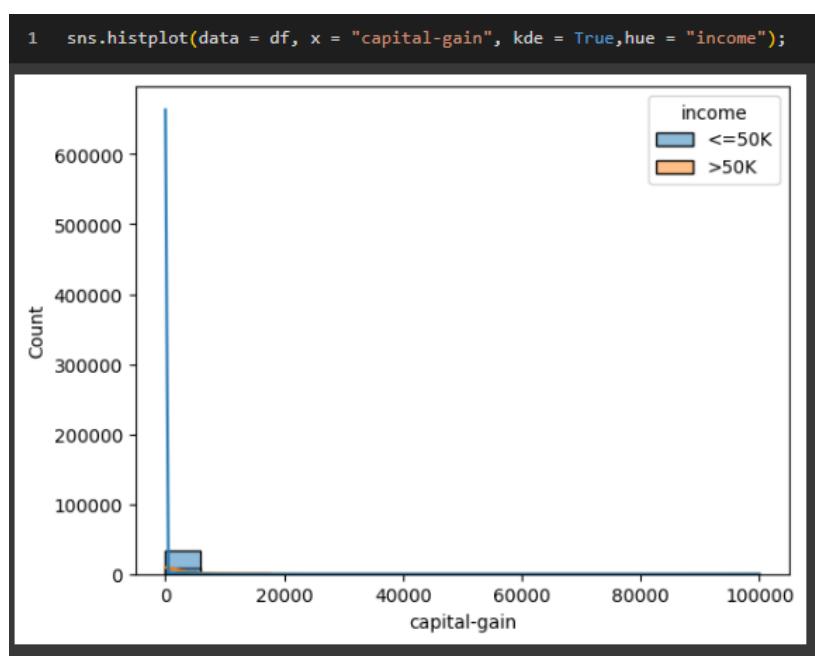
Değişkenlerin dağılımlarını görselleştirilmiş için histogramlar oluşturulmuştur.



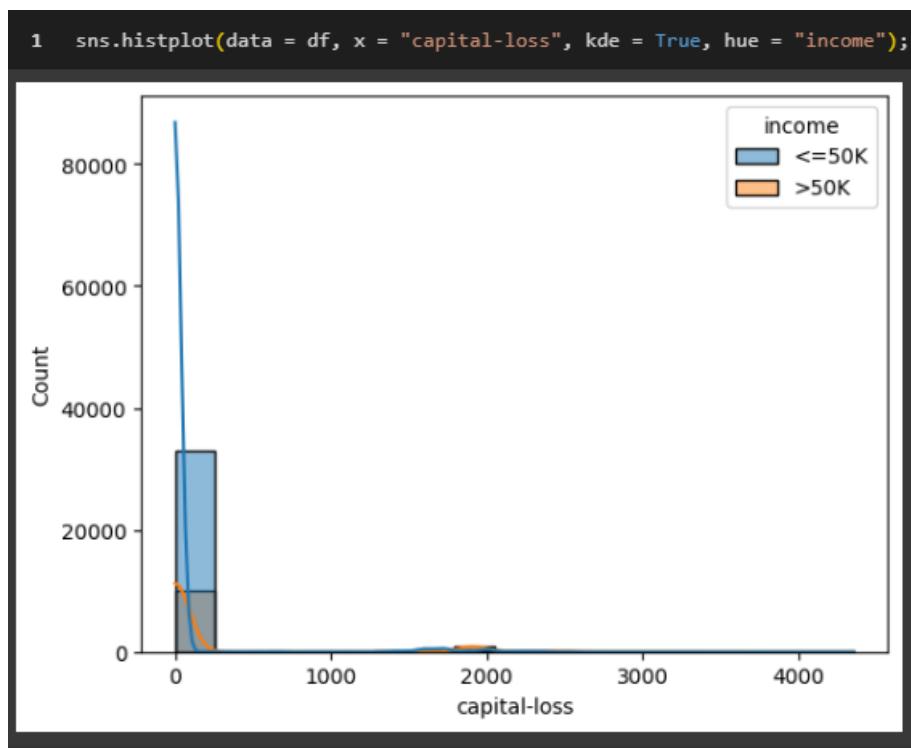
- Gelir dağılımına göre, düşük gelirli bireyler (" $\leq 50K$ ") daha genç yaşlarda yoğunlaşmaktadır. Özellikle 20-40 yaş arası bireylerde bu grup daha fazladır.
- Daha yüksek gelir grubundaki bireyler (" $> 50K$ "), 30 yaş sonrasında artmaya başlamış ve yoğunlukları 40-60 yaş aralığında daha belirgin hale gelmiştir.
- Bu durum, yaşın gelir üzerindeki etkisini vurgulamakta ve yaşı ilerledikçe gelir seviyesinin artabileceğini işaret etmektedir.



- Düşük gelirli bireyler (" $\leq 50K$ "), eğitim süresi 10 yıl civarında yoğunlaşmaktadır. Bu, genelde lise mezuniyetine denk gelen bir eğitim seviyesidir.
- Daha yüksek gelirli bireyler (" $> 50K$ "), eğitim süresi 13 yıl ve üzerinde daha fazla temsil edilmektedir. Bu, genelde lisans ve üzeri eğitime denk gelmektedir.
- Eğitim süresinin artmasıyla yüksek gelir grubuna dahil olma olasılığının arttığı gözlemlenmiştir.



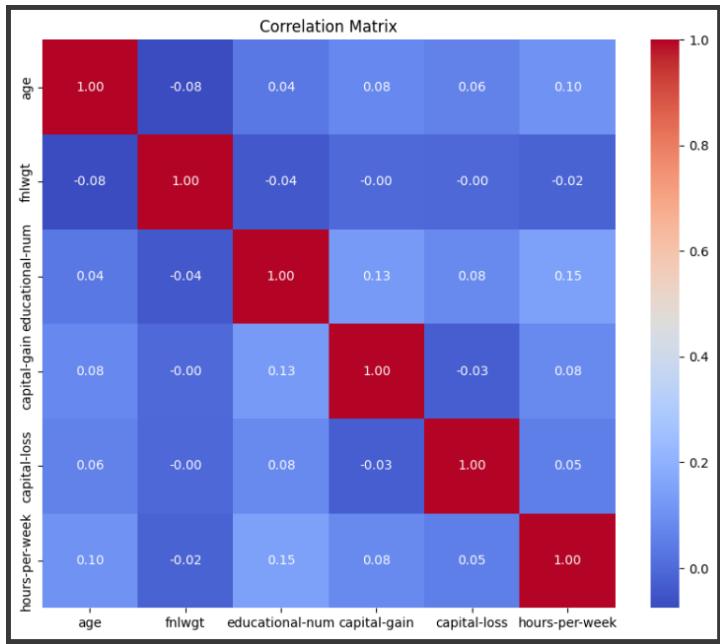
- Düşük gelir grubundaki bireylerin büyük bir çoğunluğu için sermaye kazancı sıfırdır.
- Daha yüksek gelir grubundaki bireylerde sermaye kazancı sıfır olmayan bireylerin sayısında artış gözlemlenmiştir. Ancak, yüksek değerlerde (örneğin, 50000 ve üzeri) bu artışın belirgin olduğu söylenebilir.
- Sermaye kazancına sahip bireylerin genellikle daha yüksek gelir grubunda olduğu anlaşılmaktadır.



- Benzer şekilde, düşük gelir grubundaki bireylerin çoğunlığında sermaye kaybı sıfırdır.
- Daha yüksek gelir grubundaki bireylerde düşük seviyelerde de olsa sermaye kaybı gözlemlenmiştir. Ancak, bu kayıplar genelde yüksek değerlerde daha dikkat çekicidir.
- Sermaye kaybının varlığı, daha yüksek gelir seviyelerine sahip bireylerde daha belirgin hale gelmektedir.

Aşağıdaki kod ile, veri setindeki sayısal değişkenler arasındaki ilişkiler korelasyon matrisi kullanılarak analiz edilmiştir. Korelasyon katsayıları, değişkenler arasındaki doğrusal ilişkileri temsil eder ve -1 ile 1 arasında değer alır. Pozitif değerler pozitif doğrusal ilişkiyi, negatif değerler negatif doğrusal ilişkiyi, 0'a yakın değerler ise ilişkisizlik durumunu gösterir.

```
▶ 1 # Compute the correlation matrix for numerical columns
 2 correlation_matrix = df_numerical.corr()
 3
 4 # Plot the correlation matrix as a heatmap
 5 plt.figure(figsize=(10, 8))
 6 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
 7 plt.title('Correlation Matrix')
 8 plt.show()
```



Yaş (Age) ile Diğer Değişkenler:

- Yaşın "hours-per-week" (haftalık çalışma süresi) ile pozitif bir korelasyona (**0.10**) sahip olduğu gözlemlenmiştir. Bu, yaş arttıkça çalışma saatlerinde hafif bir artış olabileceğini gösterir.
- Diğer değişkenlerle olan korelasyon değerleri çok düşüktür ve anlamlı bir ilişki göstermemektedir.

Fnlwgt (Final Weight) ile Diğer Değişkenler:

- Fnlwgt değişkeninin diğer değişkenlerle olan korelasyonu sıfır çok yakındır. Bu, bu değişkenin diğer değişkenlerle anlamlı bir ilişkisinin olmadığını göstermektedir.

Educational-num (Eğitim Süresi):

- Eğitim süresinin "hours-per-week" ile pozitif bir korelasyona (**0.15**) sahip olduğu gözlemlenmiştir. Bu, eğitim seviyesi arttıkça bireylerin haftalık çalışma saatlerinin hafifçe artabileceğini göstermektedir.
- Ayrıca, eğitim süresinin "capital-gain" ile (**0.13**) pozitif bir korelasyonu bulunmaktadır. Bu, eğitim seviyesi arttıkça sermaye kazancının artabileceğini işaret eder.

Capital-gain (Sermaye Kazancı) ve Capital-loss (Sermaye Kaybı):

- Sermaye kazancı ile sermaye kaybı arasında belirgin bir korelasyon görülmemektedir. Bu, her iki değişkenin birbirinden bağımsız olarak değiştigini gösterir.
- Diğer değişkenlerle olan ilişkiler oldukça zayıftır.

Hours-per-week (Haftalık Çalışma Süresi):

- Haftalık çalışma süresi, eğitim süresi (**0.15**) ile en güçlü ilişkiye sahiptir. Bu, bireylerin eğitim düzeyinin artmasıyla çalışma sürelerinin bir miktar arttığını gösterir.
- Diğer değişkenlerle korelasyon değerleri çok düşüktür.

3.7 Aykırı Değer Analizi

Veri ön işleme kısmında aykırı değerlerin tespiti, veri kalitesini artırmak ve model performansını iyileştirmek için önemli bir işlemdir. Aykırı değerler, veri setindeki diğer gözlemlerle uyuşmayan, alışılmışın dışındaki değerlerdir ve genellikle hatalı ölçümleri ve veri girişindeki hataları temsil eder. Aykırı değerlerin tespiti için birçok yöntem kullanılabilir. Bu projede IQR (Interquartile Range) yöntemi kullanıldı. Bu yöntemde verilerin çeyrekler aralığı hesaplanarak, alt sınırın altında ve üst sınırın üstündeki değerler aykırı kabul edilir. Kutu grafikleri(Boxplot) ile tespit edilen aykırı değerler görselleştirilmiştir. İlk olarak age verilerindeki aykırılıklar tespit edilmiştir. Aşağıdaki kod ile çeyreklik ve sınır hesaplamaları, verileri boxplot ile görselleştirilmesi ve aykırı değerler çıkarılmadan önceki veri sayısının tespiti amaçlanmıştır.

```
df_age = df['age']

# Quantile değerlerin belirlenmesi.
Q1 = df_age.quantile(0.25)
Q3 = df_age.quantile(0.75)

print("Q1:",Q1)
print("Q3:",Q3)

# IQR değerin belirlenmesi.
IQR = Q3-Q1
print("IQR:",IQR)

# Alt ve üst sınırların belirlenmesi.
lower_fence = Q1 - 1.5*IQR
upper_fence = Q3 + 1.5*IQR

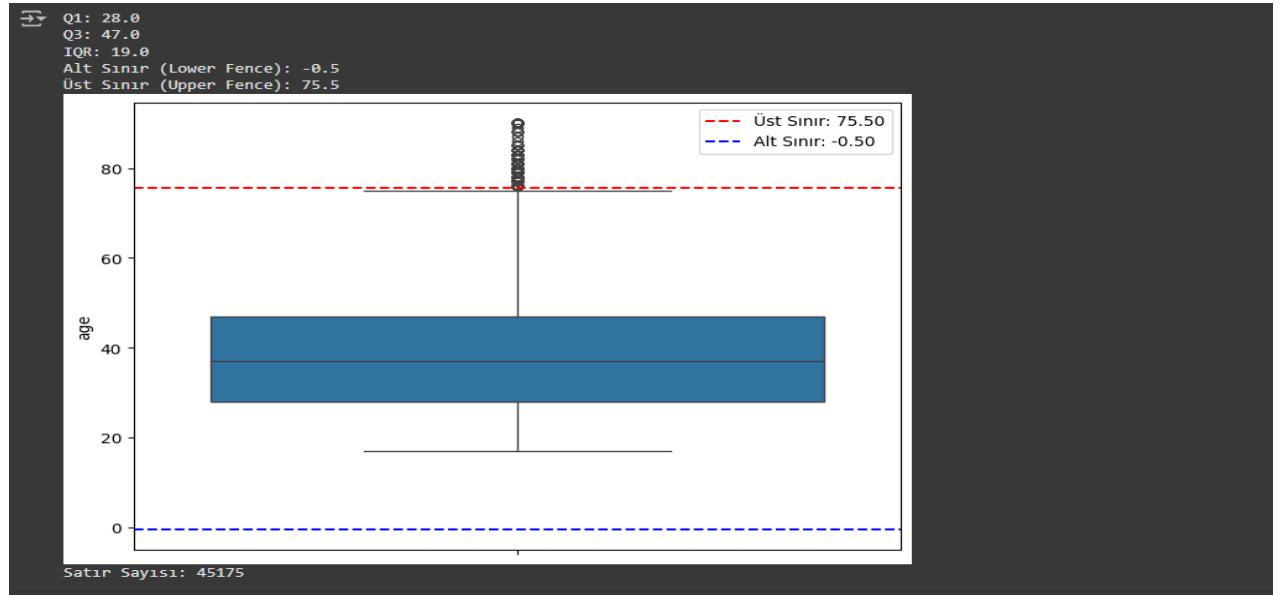
print("Alt Sınır (Lower Fence):", lower_fence)
print("Üst Sınır (Upper Fence):", upper_fence)

df = df.copy()
# Bir değişkendeki IQR'a göre aykırı gözlemleri boxplot kullanarak görselleştirelim.
plt.figure(figsize = (8, 6))
sns.boxplot(data = df,
            y = df["age"],
            orient = "v");

plt.axhline(y=upper_fence, color='r', linestyle='--', label=f"Üst Sınır: {upper_fence:.2f}")
plt.axhline(y=lower_fence, color='b', linestyle='--', label=f"Alt Sınır: {lower_fence:.2f}")
plt.legend()
plt.show()

print(f"Satır Sayısı: {len(df)})
```

Kod çalıştırıldığında sonuç şu şekildedir ;

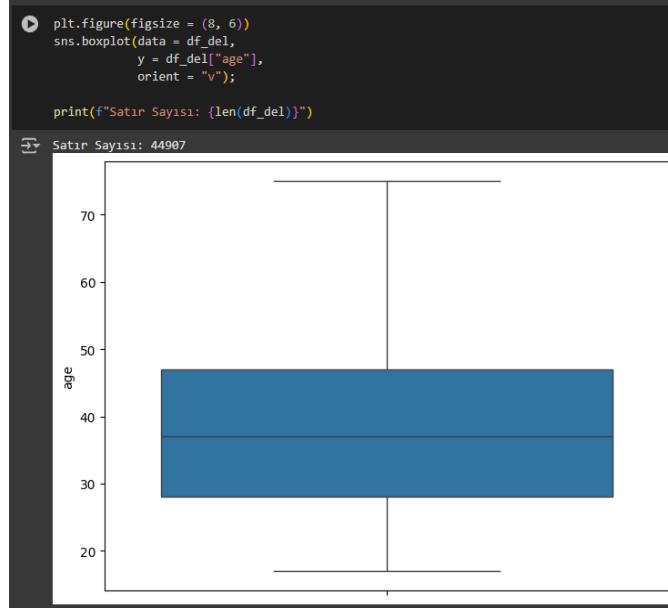


Oluşan boxplot ile üst sınırı aşan aykırı değerler tespit edilmiştir. Aşağıdaki kod bu değerleri görmemizi sağlar. Sonrasında bu tespit edilen üst sınırı aşan değerler veri setinden silinip aykırı

değerler çıkarılmıştır. Bu işlem ile veri sayımız 45175'ten 44907'e düşmüştür.

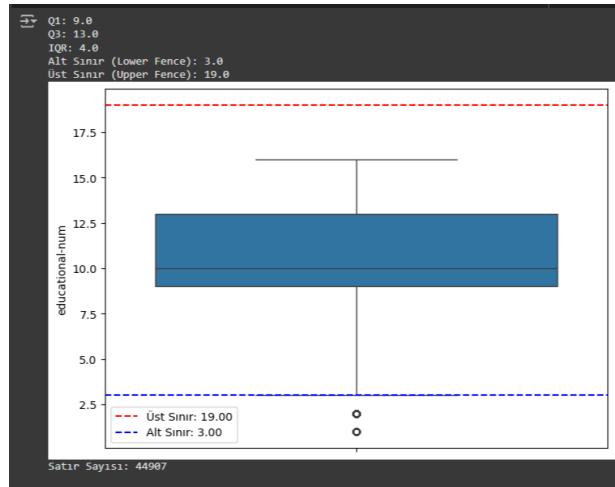
```
[ ] # Upper_fence üzerinde kalan aykırı gözlemlerin index değerlerini, daha sonra kullanmak üzere bir değişkende tutabiliyoruz.  
outlier_idx = df_age[df_age > upper_fence].index  
  
[ ] outlier_idx  
  
[ ] Index([ 234, 899, 951, 1034, 1079, 1373, 1398, 2085, 2290, 2294,  
...  
45002, 45229, 45429, 45875, 47311, 48095, 48136, 48558, 48648, 48740],  
dtype='int64', length=268)  
  
[ ] df_del = df[~(df_age > upper_fence)]  
df_del  
  
print(f"Satır Sayısı: {len(df_del)}")  
Satır Sayısı: 44907
```

Kod aykırı değerlerden arındırılmış age verisinin boxplot grafiğini vermektedir.



Aynı işlemler educational-num sütununa uygulanmıştır. Çeyreklik ve sınır hesaplamaları, verileri boxplot ile görselleştirilmesi ve aykırı değerler çıkarılmadan önceki veri sayısı aşağıdaki gibidir:

Boxplot grafiğinde alt sınırın altında kalan aykırı değerler tespit edilmiştir, bu değerleri görmek için ve silmek için yazılan kodlar şu şekildedir:



```
[ ] # lower_fence altında kalan aykırı gözlemlerin index değerlerini, daha sonra kullanmak üzere bir değişkende tutabilmiriz.
outlier_idx = df_educational_num[df_educational_num < lower_fence].index

[ ] outlier_idx
[ ] Index([ 323, 519, 779, 818, 890, 961, 1059, 1186, 1279, 1390,
... 48146, 48154, 48316, 48325, 48505, 48587, 48640, 48684, 48706, 48713],
dtype='int64', length=290)

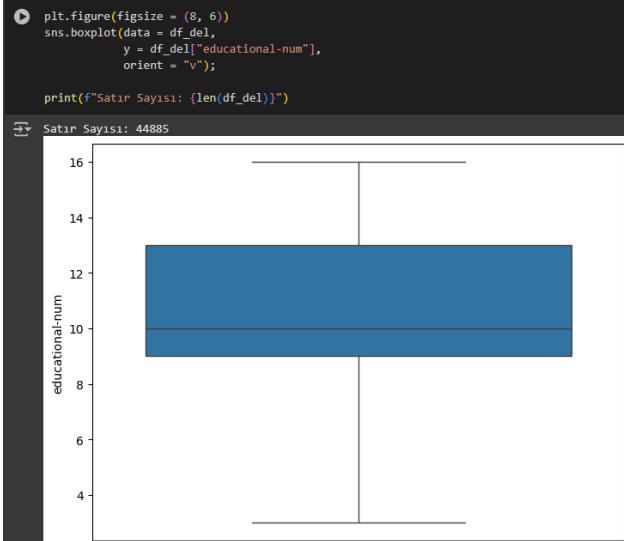
[ ] df_del = df[~(df_educational_num < lower_fence)]

[ ] df_del
print(f"Satır Sayısı: {len(df_del)}")

[ ] Satır Sayısı: 44885
```

Aykırı olan değerler silindiğinden sonra 44907 olan veri sayısı 44885'e düşmüştür.

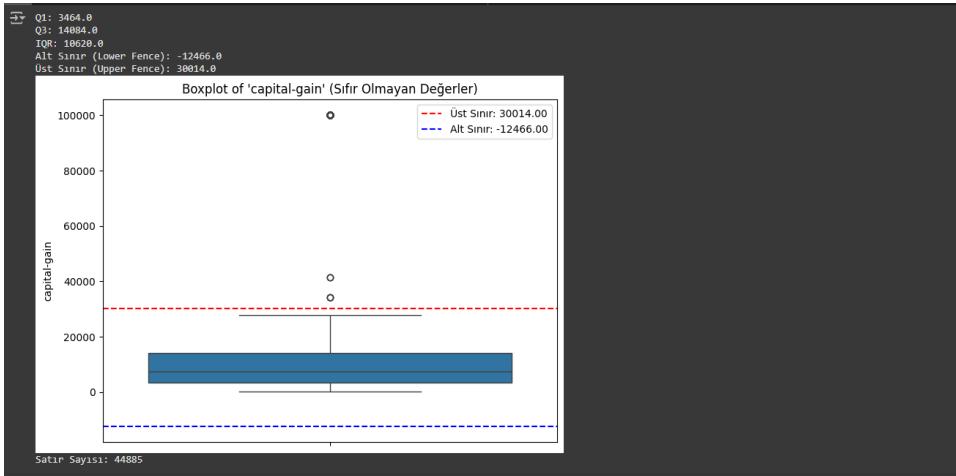
Aykırı veriden temizlenmiş verinin boxplot grafiği şu şekildedir:



Sıradaki aykırı değerlerinden temizlenecek değişken ise capital-gain'dir. Capital-gain veri setinin çoğunluğu sıfır değerinden oluştugu için IQR ve çeyreklikler sıfır değerini almaktaydı. Bu sağlıklı bir analizi etkileyen bir durum olduğu için aykırı değer tespitinde sıfır olan veriler çıkartılarak inceleme yapılmıştır. Bu verilerin boxplot grafiği çizilmiştir. Sıfırlardan ayılanmış verilerin çeyreklik , sınırlar ve boxplot oluşturma kodu aşağıdaki gibidir:

```
[ ] # Sıfır olmayan 'capital-gain' değerlerini filtreleme
df_capital_gain = df['capital-gain']
df_non_zero = df_capital_gain[df_capital_gain > 0]
# Quantile değerlerin belirlenmesi
Q1 = df_non_zero.quantile(0.25)
Q3 = df_non_zero.quantile(0.75)
print('Q1:', Q1)
print('Q3:', Q3)
# IQR değerinin belirlenmesi
IQR = Q3 - Q1
print("IQR:", IQR)
# Alt ve Üst sınırların belirlenmesi
lower_fence = Q1 - 1.5 * IQR
upper_fence = Q3 + 1.5 * IQR
print("Alt Sınır (Lower Fence):", lower_fence)
print("Üst Sınır (Upper Fence):", upper_fence)
# Sıfır olmayan 'capital-gain' değerlerini filtreleme
df_non_zero = df[df['capital-gain'] > 0]
# Q1, Q3 ve IQR hesaplama
Q1_non_zero = df_non_zero['capital-gain'].quantile(0.25)
Q3_non_zero = df_non_zero['capital-gain'].quantile(0.75)
IQR_non_zero = Q3_non_zero - Q1_non_zero
# Alt ve üst sınırların hesaplanması
lower_fence_non_zero = Q1_non_zero - 1.5 * IQR_non_zero
upper_fence_non_zero = Q3_non_zero + 1.5 * IQR_non_zero
# Boxplot ile görselleştirme
plt.figure(figsize=(8, 6))
sns.boxplot(data=df_non_zero, y='capital-gain', orient='v')
plt.title("Boxplot of 'capital-gain' (Sıfır Olmayan Değerler)")
plt.axhline(y=upper_fence_non_zero, color='r', linestyle='--', label=f"Üst Sınır: {upper_fence_non_zero:.2f}")
plt.axhline(y=lower_fence_non_zero, color='b', linestyle='--', label=f"Alt Sınır: {lower_fence_non_zero:.2f}")
plt.legend()
plt.show()
print(f"Satır Sayısı: {len(df_del)})")
```

Kod çıktısı şu şekildedir:



Üst sınırı aşan değerler tespit edilmiştir aşağıdaki kod ile bu değerler listelenmiştir.

```
[ ] # Upper_fence üzerinde kalan aykırı gözlemlerin index değerlerini, daha sonra kullanmak üzere bir değişkende tutabiliriz.
outlier_idx = df_non_zero[df_non_zero['capital-gain'] > upper_fence].index

outlier_idx
```

```
Index([ 346, 357, 418, 692, 702, 882, 992, 1131, 1154, 1157,
       ...,
       45916, 46087, 46525, 46777, 47194, 47392, 48109, 48253, 48519, 48799],
      dtype='int64', length=236)
```

Tespit edilen sıfırdan farklı aykırı değerler veriden çıkartılmıştır. Sonrasında ayıklanmış veri sayısı yazdırılmıştır. Veri sayısı 44885'ten 44650'ye inmiştir.

```
# Aykırı değerlerin index'lerini bulma
outlier_idx = df_non_zero[df_non_zero['capital-gain'] > upper_fence].index

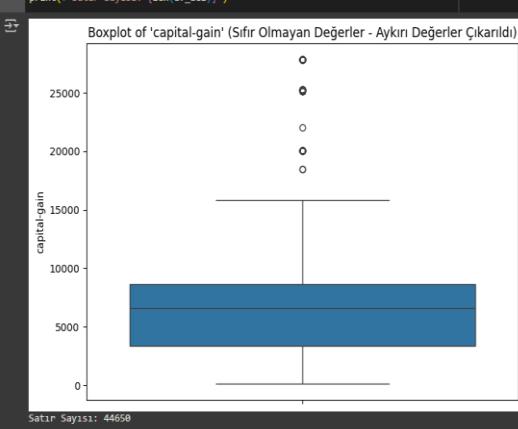
# Aykırı değerleri çıkararak df_del'i güncelleme
df_del = df_del.drop(index=outlier_idx)

# Güncellmiş veri setinin boyutunu kontrol etme
print("Aykırı değerler çıkarıldıkten sonraki veri seti boyutu: {}").format(df_del.shape[0])
```

Aykırı veriden temizlenmiş son veri boxplot üzerinde görselleştirilmiştir. Şekilde aşağıda gösterilmiştir. Bir kısım veri dışarıda kalmıştır.

```
# Sıfır olmayan 'capital-gain' değerlerini içeren df_del'i filtreleme
df_non_zero = df_del[df_del['capital-gain'] > 0]
# Boxplot ile sıfır olmayan değerlerin görselleştirilmesi
plt.figure(figsize=(8, 6))
sns.boxplot(data=df_non_zero, y='capital-gain', orient='v')
plt.title("Boxplot of 'capital-gain' (Sıfır Olmayan Değerler - Aykırı Değerler Çıkarıldı)")
plt.show()

print("Satır Sayısı: {}".format(len(df_del)))
```



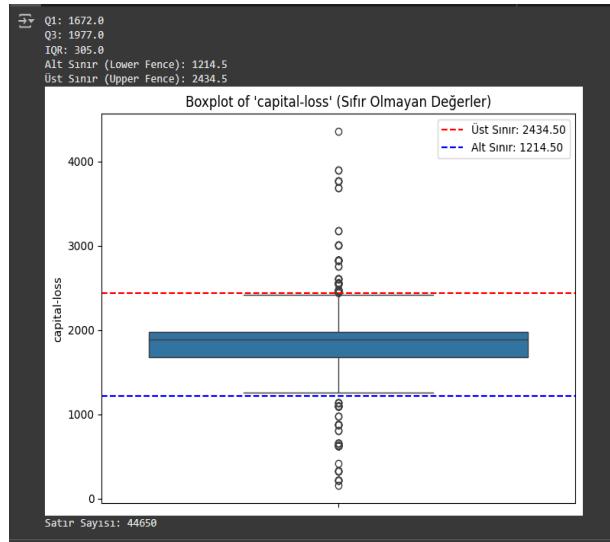
Son olarak capital-loss verilerinde aykırılık analizi yapılmıştır. Sıfırdan büyük değerlere analiz yapılarak çeyreklikler, alt-üst sınırlar hesaplanmıştır. Hesaplanan sınırlar doğrultusunda Boxplot grafiği ile görselleştirme yapılmıştır ve aykırı değerler görselleştirilmiştir. Sonrasında aykırı değerden temizlenmeden önceki veri sayısı yazdırılmıştır. Bu işlemleri yapan kod aşağıdaki gibidir:

```
# Sıfır olmayan 'capital-loss' değerlerini filtreleme
df_capital_loss = df['capital-loss']
df_non_zero = df_capital_loss[df_capital_loss > 0]
# Quantile değerlerin belirlenmesi
Q1 = df_non_zero.quantile(0.25)
Q3 = df_non_zero.quantile(0.75)
print("Q1:", Q1)
print("Q3:", Q3)
# IQR değerinin belirlenmesi
IQR = Q3 - Q1
print("IQR:", IQR)
# Alt ve Üst sınırların belirlenmesi
lower_fence = Q1 - 1.5 * IQR
upper_fence = Q3 + 1.5 * IQR
print("Alt Sınır (Lower Fence):", lower_fence)
print("Üst Sınır (Upper Fence):", upper_fence)

# Sıfır olmayan 'capital-loss' değerlerini filtreleme
df_non_zero = df[df['capital-loss'] > 0]
# Q1, Q3 ve IQR hesaplama
Q1_non_zero = df_non_zero['capital-loss'].quantile(0.25)
Q3_non_zero = df_non_zero['capital-loss'].quantile(0.75)
IQR_non_zero = Q3_non_zero - Q1_non_zero
# Alt ve üst sınırların hesaplanması
lower_fence_non_zero = Q1_non_zero - 1.5 * IQR_non_zero
upper_fence_non_zero = Q3_non_zero + 1.5 * IQR_non_zero
# Boxplot ile görselleştirme
plt.figure(figsize=(8, 6))
sns.boxplot(data=df_non_zero, y="capital-loss", orient='v')
plt.title("Boxplot of 'capital-loss' (Sıfır Olmayan Değerler)")
plt.axhline(y=upper_fence_non_zero, color='r', linestyle='--', label=f"Üst Sınır: {upper_fence_non_zero:.2f}")
plt.axhline(y=lower_fence_non_zero, color='b', linestyle='--', label=f"Alt Sınır: {lower_fence_non_zero:.2f}")
plt.legend()
plt.show()

print(f"Satır Sayısı: {len(df)}")
```

Kod çıktısı sonucunda capital-loss değerlerinde alt ve üst sınırı aşan aykırı değerlerin olduğu tespit edilmiştir.



Aykırı veriler liste halinde incelenmiştir sonrasında bu değerler capital-loss verisinden temizlenmiştir. Üst ve alt sınırların dışında kalan değerler silindiğten sonra veri sayısı 44650'den 44509'a inmiştir.

```
[ ] # Upper_fence üzerinde kalan ve lower_fence altında kalan aykırı gözlemlerin index değerlerini, daha sonra kullanmak üzere bir değişkende tutabiliyoruz.
outlier_idx = df_capital_loss[(df_capital_loss > upper_fence) | (df_capital_loss < lower_fence)].index

outlier_idx
→ Index([    0,     1,     2,     3,     5,     7,     8,     9,    10,    11,
...     48832, 48833, 48834, 48835, 48836, 48837, 48838, 48839, 48840, 48841],
      dtype='int64', length=43177)

▶ # Sıfır olmayan 'capital-loss' değerlerini filtreleme
df_non_zero = df_del[df_del['capital-loss'] > 0]
# Q1, Q3 ve IQR hesaplama
Q1_non_zero = df_non_zero['capital-loss'].quantile(0.25)
Q3_non_zero = df_non_zero['capital-loss'].quantile(0.75)
IQR_non_zero = Q3_non_zero - Q1_non_zero
# Alt ve üst sınırların hesaplanması
lower_fence_non_zero = Q1_non_zero - 1.5 * IQR_non_zero
upper_fence_non_zero = Q3_non_zero + 1.5 * IQR_non_zero
# Aykırı değerlerin index'lerini bulma (alt sınırın altı ve üst sınırın üstü)
outlier_idx = df_non_zero[
    (df_non_zero['capital-loss'] > upper_fence_non_zero) |
    (df_non_zero['capital-loss'] < lower_fence_non_zero)
].index

# Aykırı değerleri çıkararak df_del'i güncelleme
df_del = df_del.drop(index=outlier_idx)

# Güncellenmiş veri setinin boyutunu kontrol etme
print(f"Aykırı değerler çıkarıldıkten sonraki veri seti boyutu: {df_del.shape[0]}")

→ Aykırı değerler çıkarıldıkten sonraki veri seti boyutu: 44509
Satır Sayısı: 44509
```

3.8 Veri Ölçekleme

3.8.1 Normalizasyon

Normalizasyon, veri ön işleme aşamasında, değişkenlerin belirli bir aralığa (genellikle 0 ile 1 arasına) ölçeklenmesi adımdır. Normalizasyon için **sklearn.preprocessing** kütüphanesinin **MinMaxScaler** sınıfı import edilmiştir. Veride age katagorisi dışındaki nümerik veriler normalize edilmiştir. Age verilerinin dışında tutulmasının nedeni Age değerlerinin çok uç değerleri ifade etmemesidir. Aşağıdaki kod ile age dışındaki nümerik sütunların normalizasyonu yapılmıştır:

```
[ ] from sklearn.preprocessing import MinMaxScaler

# Min-Max Scaler nesnesini oluşturma
scaler = MinMaxScaler()

# Sayısal sütunları seçme
numeric_columns = df_del.select_dtypes(include=['float64', 'int64']).columns
numeric_columns = numeric_columns.drop('age') # 'age' sütununu çıkar

# Min-Max Scaling uygulama
df_del = df_del.copy()
df_del[numeric_columns] = scaler.fit_transform(df_del[numeric_columns])

# Sonuçları gösterme
print("Min-Max Scaling uygulandıktan sonra veri seti:")
print(df_del.head())

# Orijinal ve ölçeklendirilmiş veriyi kontrol etmek için
print("Orijinal Veri:")
print(df_del[numeric_columns].head())

print("Normalize Edilmiş Veri:")
print(df_del[numeric_columns].head())

print(f"Satır Sayısı: {len(df_del)}")
```

Sütunların normalize edilmiş veriler ile güncellenmiş halinin çıktısı aşağıdaki gibidir:

```
↳ Min-Max Scaling uygulandıktan sonra veri seti:
   age  workclass  fnlwgt    education  educational-num \
0    25    Private  0.144430      11th      0.307692
1    38    Private  0.051677     HS-grad      0.461538
2    28  Local-gov  0.219011  Assoc-acdm      0.692308
3    44    Private  0.099418  Some-college      0.538462
5    34    Private  0.125398      10th      0.230769

      marital-status    occupation  relationship    race gender \
0  Never-married  Machine-op-inspct  Own-child  Black  Male
1 Married-civ-spouse  Farming-fishing    Husband  White  Male
2 Married-civ-spouse  Protective-serv    Husband  White  Male
3 Married-civ-spouse  Machine-op-inspct    Husband  Black  Male
5 Never-married      Other-service  Not-in-family  White  Male

  capital-gain  capital-loss  hours-per-week native-country income
0      0.000000        0.0       0.397959 United-States <=50K
1      0.000000        0.0       0.500000 United-States <=50K
2      0.000000        0.0       0.397959 United-States >50K
3      0.276269        0.0       0.397959 United-States >50K
5      0.000000        0.0       0.295918 United-States <=50K

Orijinal Veri:
   fnlwgt  educational-num  capital-gain  capital-loss  hours-per-week
0  0.144430      0.307692      0.000000        0.0       0.397959
1  0.051677      0.461538      0.000000        0.0       0.500000
2  0.219011      0.692308      0.000000        0.0       0.397959
3  0.099418      0.538462      0.276269        0.0       0.397959
5  0.125398      0.230769      0.000000        0.0       0.295918

Normalize Edilmiş Veri:
   fnlwgt  educational-num  capital-gain  capital-loss  hours-per-week
0  0.144430      0.307692      0.000000        0.0       0.397959
1  0.051677      0.461538      0.000000        0.0       0.500000
2  0.219011      0.692308      0.000000        0.0       0.397959
3  0.099418      0.538462      0.276269        0.0       0.397959
5  0.125398      0.230769      0.000000        0.0       0.295918

Satır Sayısı: 44509
```

3.9 Veri Dönüşümü

3.9.1 Label Encoding Dönüşümü

Encoding işleminde katagorik olan veriler nümerik hale çevrilir. İlk olarak katagorik değişken sahip olan sütunların değişken isimleri belirlenmektedir. Her kategorinin değişkenleri ilk adımda kendi içinde mantık sırasına göre sıralama işlemi yapıldı. (örn: education için okul öncesinden(preschool) prof.(prof-school)'a sıralama gibi)

```
from pandas.api.types import CategoricalDtype

# Sıralama mantığını tanımlıyoruz

workclass_order = [
    "Never-worked", "Without-pay", "Federal-gov", "Local-gov", "State-gov",
    "Self-emp-not-inc", "Self-emp-inc", "Private"
]

education_order = [
    "Preschool", "1st-4th", "5th-6th", "7th-8th", "9th", "10th", "11th", "12th",
    "HS-grad", "Some-college", "Assoc-voc", "Assoc-acdm", "Bachelors", "Masters", "Doctorate", "Prof-school"
]

marital_status_order = [
    "Never-married", "Separated", "Divorced", "Widowed", "Married-spouse-absent",
    "Married-AF-spouse", "Married-civ-spouse"
]

occupation_order = [
    "Priv-house-serv", "Other-service", "Handlers-cleaners", "Farming-fishing", "Machine-op-inspct", "Transport-moving",
    "Craft-repair", "Adm-clerical", "Sales", "Tech-support", "Protective-serv", "Prof-specialty", "Exec-managerial", "Armed-Forces"
]

relationship_order = [
    "Other-relative", "Not-in-family", "Own-child", "Unmarried", "Wife", "Husband"
]

race_order = ["Other", "Amer-Indian-Eskimo", "Asian-Pac-Islander", "Black", "White"]
gender_order = ["Female", "Male"]
income_order = ["<=50K", ">50K"]
```

Aşağıdaki kod ile listedeki sıraya göre kategorik değişkenlere sayılar atanır. Ancak bu aşamada tüm kategorik sütunlara işlem yapılmamaktadır. native-country kategorisi ayrı tutularak encoding işlemi yapılır. Atamayı yaparak verinin ilk 6 satırı yazdırılmıştır:

```
[ ] from pandas.api.types import CategoricalDtype  
  
# Sıralama mantığını yukarıda kategorik verilerin analizinde tanımlamıştık.  
  
# Kategorik sütunlara sıralamaları uygulama ve encoding yapma  
df_del['workclass'] = df_del['workclass'].astype(CategoricalDtype(categories=workclass_order, ordered=True)).cat.codes  
df_del['education'] = df_del['education'].astype(CategoricalDtype(categories=education_order, ordered=True)).cat.codes  
df_del['marital-status'] = df_del['marital-status'].astype(CategoricalDtype(categories=marital_status_order, ordered=True)).cat.codes  
df_del['occupation'] = df_del['occupation'].astype(CategoricalDtype(categories=occupation_order, ordered=True)).cat.codes  
df_del['relationship'] = df_del['relationship'].astype(CategoricalDtype(categories=relationship_order, ordered=True)).cat.codes  
df_del['race'] = df_del['race'].astype(CategoricalDtype(categories=race_order, ordered=True)).cat.codes  
df_del['gender'] = df_del['gender'].astype(CategoricalDtype(categories=gender_order, ordered=True)).cat.codes  
df_del['income'] = df_del['income'].astype(CategoricalDtype(categories=income_order, ordered=True)).cat.codes  
  
# Sonucu kontrol etme  
print(df_del.head())  
  
print(f"Satır Sayısı: {len(df_del)}")
```

Kodun çıktısında katagorik olan sütunların birçoğu (native-country hariç) nümerik olmuştur. Düzenlenmiş veri şu şekilde görülmektedir:

```
   age  workclass    fnlwgt  education  educational-num  marital-status  \n  
0  25      7  0.144430          6      0.307692          0  
1  38      7  0.051677          8      0.461538          6  
2  28      3  0.219011         11      0.692308          6  
3  44      7  0.099418          9      0.538462          6  
5  34      7  0.125398          5      0.230769          0  
  
  occupation  relationship  race  gender  capital-gain  capital-loss  \n  
0           4              2    3     1  0.000000        0.0  
1           3              5    4     1  0.000000        0.0  
2          10              5    4     1  0.000000        0.0  
3           4              5    3     1  0.276269        0.0  
5           1              1    4     1  0.000000        0.0  
  
hours-per-week  native-country  income  
0       0.397959  United-States  0  
1       0.500000  United-States  0  
2       0.397959  United-States  1  
3       0.397959  United-States  1  
5       0.295918  United-States  0  
Satır Sayısı: 44509
```

Her bir kategorik değişken için toplam gözlem sayısını görmek aşağıdaki kod yazılmıştır:

```
[ ] #Her bir kategorik değişken için toplam gözlem sayısını value_counts() ile bul.  
for col in categorical_features:  
    print(f'{df_del[col].value_counts()}\n', 5*****)  
  
print(f"Satır Sayısı: {len(df_del)}")
```

Kategorik sütunlardaki değişkenlerin toplam gözlem sayıları:

workclass	occupation	native-country
7 32781	6 5952	United-States 40834
5 3721	12 5874	Mexico 767
3 3078	11 5867	Philippines 272
4 1929	7 5581	Germany 192
6 1581	8 5345	Puerto-Rico 166
2 1398	1 4719	Canada 161
1 21	4 2912	India 142
Name: count, dtype: int64	5 2296	El-Salvador 128
*****	2 2011	Cuba 128
education	3 1422	England 117
8 14700	9 1411	China 109
9 9852	10 970	Jamaica 102
12 7473	8 215	South 100
13 2470	13 14	Italy 96
10 1948	Name: count, dtype: int64	Japan 88
6 1614	*****	Dominican-Republic 84
11 1502	relationship	Columbia 80
5 1215	5 18355	Poland 80
3 819	1 11494	Vietnam 79
15 711	2 6589	Guatemala 76
4 673	3 4684	Haiti 64
7 575	4 2069	Iran 56
14 513	0 1318	Portugal 55
2 444	Name: count, dtype: int64	Taiwan 54
Name: count, dtype: int64	*****	Greece 48
*****	race	Nicaragua 46
marital-status	4 38295	Peru 45
6 20712	3 4178	Ecuador 42
0 14426	2 1271	Ireland 36
2 6193	1 431	France 36
1 1381	0 334	Thailand 28
3 1237	Name: count, dtype: int64	Hong 27
4 529	*****	Trinidad&Tobago 26
5 31	gender	Cambodia 24
Name: count, dtype: int64	1 30013	Yugoslavia 23
*****	0 14496	Outlying-US(GUAM-USVI-etc) 22
Name: count, dtype: int64	Name: count, dtype: int64	Scotland 20
*****	Native-country için kategorik dönüşüm ve encoding	Laos 20
native-country_order = sorted(df_del['native-country'].unique())	native_country_dtype = CategoricalDtype(categories=native_country_order, ordered=True)	Hungary 18
# 4. Native-country için kategorik dönüşüm ve encoding	df_del['native-country'] = df_del['native-country'].astype(native_country_dtype).cat.codes	Honduras 17
# 5. Encoding sonucunda ülke ve kod eşleştirme	native_country_mapping = {code: country for code, country in enumerate(native_country_order)}	Holand-Netherlands 1
# 6. Sonuçları yazdırma		
print("Encoding Sonucu:")		
for code, country in native_country_mapping.items():		
print(f'{country}: {code}')		
# 7. Güncellenmiş sütunu kontrol etme		
print("\nGüncellenmiş df_del 'native-country' sütunu:")		
print(df_del['native-country'].head())		

Son encoding aşaması olarak native country kısmını ayrı yaptık nedeni ise; 40 tane ülke adı olduğu için 100'ün altında gözlem sayısı olan ülkeleri other olarak tanımladık. Sonrasında ise ülke sayı eşleşmelerini görmek için aşağıdaki gibi kod yazılmıştır:

```
[ ] from pandas.api.types import CategoricalDtype
# 1. Frekansları hesapla (value_counts)
counts = df_del['native-country'].value_counts()

# 2. Frekansı 100'ün altında olanları "Other" olarak değiştir
# Daha verimli map fonksiyonu kullanımı
threshold = 100
df_del['native-country'] = df_del['native-country'].map(lambda x: x if counts[x] >= threshold else 'Other')

# 3. Alfabetik sıraya göre benzersiz değerlerin sıralanması
native_country_order = sorted(df_del['native-country'].unique())

# 4. Native-country için kategorik dönüşüm ve encoding
native_country_dtype = CategoricalDtype(categories=native_country_order, ordered=True)
df_del['native-country'] = df_del['native-country'].astype(native_country_dtype).cat.codes

# 5. Encoding sonucunda ülke ve kod eşleştirme
native_country_mapping = {code: country for code, country in enumerate(native_country_order)}

# 6. Sonuçları yazdırma
print("Encoding Sonucu:")
for code, country in native_country_mapping.items():
    print(f'{country}: {code}')

# 7. Güncellenmiş sütunu kontrol etme
print("\nGüncellenmiş df_del 'native-country' sütunu:")
print(df_del['native-country'].head())
```

Ülke sayı eşleşmeleri ve veri setinin native-country sütununun ilk 6 verisi şu şekildedir:

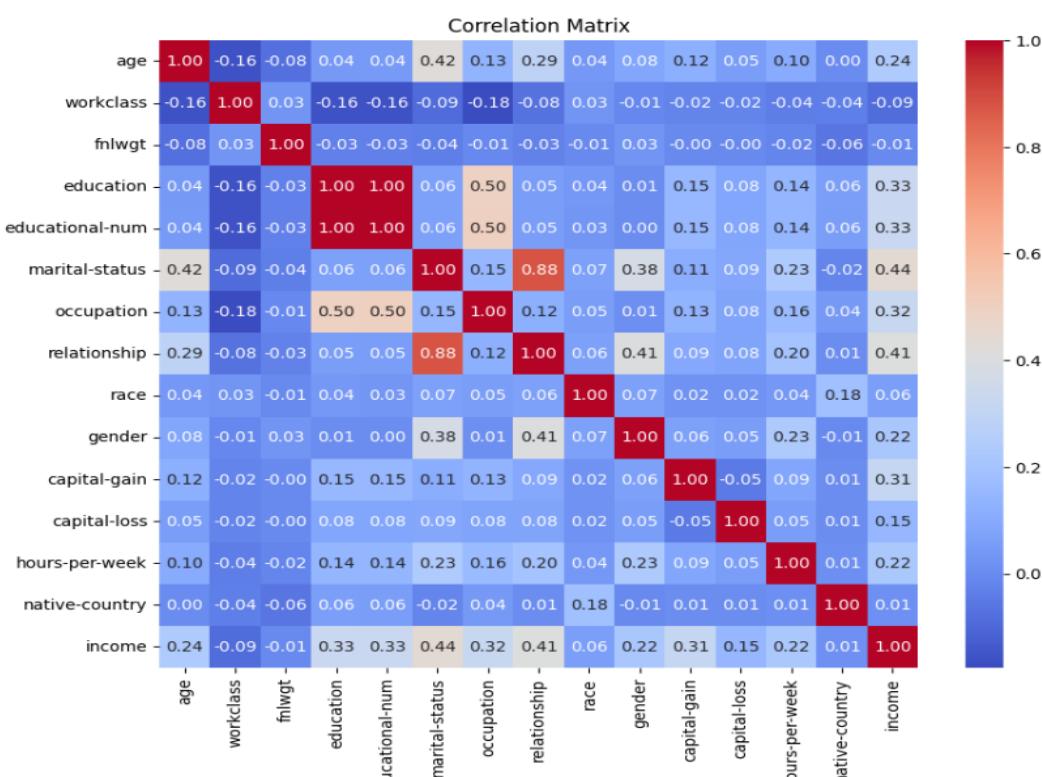
```
Encoding Sonucu:  
Canada: 0  
China: 1  
Cuba: 2  
El-Salvador: 3  
England: 4  
Germany: 5  
India: 6  
Jamaica: 7  
Mexico: 8  
Other: 9  
Philippines: 10  
Puerto-Rico: 11  
South: 12  
United-States: 13  
  
Güncellenmiş df_del 'native-country' sütunu:  
0    13  
1    13  
2    13  
3    13  
5    13  
Name: native-country, dtype: int8  
Satır Sayısı: 44509
```

Düzenlenmiş veri setinin ilk 6 satırını gösteren kod ve çıktısı aşağıdaki gibidir:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	7	0.144430	6	0.307692	0	4	2	3	1	0.000000	0.0	0.397959	13	0
1	38	7	0.051677	8	0.461538	6	3	5	4	1	0.000000	0.0	0.500000	13	0
2	28	3	0.219011	11	0.692308	6	10	5	4	1	0.000000	0.0	0.397959	13	1
3	44	7	0.099418	9	0.538462	6	4	5	3	1	0.276269	0.0	0.397959	13	1
5	34	7	0.125398	5	0.230769	0	1	1	4	1	0.000000	0.0	0.295918	13	0

3.9.2 Özelliğ İndirgeme (Feature Reduction)

Son olarak veride korelasyon matrisi ile kategoriler arası benzerliklere bakılacaktır. Matristeki benzerlik oranlarına göre yüksek benzerlikteki kategoriler veriden çıkarılacaktır yani ön işlemde sütun olarak temizleme yapılacaktır. Verinin Korelasyon matrisi aşağıdaki gibidir:



education & educational-num: Pozitif güçlü korelasyon (1.00). Eğitim seviyesi ve eğitim sayısal değeri arasında güçlü bir ilişki var.

marital-status & relationship: Pozitif güçlü korelasyon (0.88). Medeni durum ve akraba sayısal değeri arasında güçlü bir ilişki var. Bu sütunlar arasında çıkarma işlemi yaptık. Sonrasında güncellenmiş sütunlar yazdırıldı.

```
# 'marital-status' ve 'educational-num' sütunlarını çıkarma
df_del = df.drop(columns=['marital-status', 'educational-num'])

# Güncellenmiş veri setini kontrol etme
print("Güncellenmiş sütunlar:")
print(df_del.columns)

print(f"Satır Sayısı: {len(df_del)}")

→ Güncellenmiş sütunlar:
Index(['age', 'workclass', 'fnlwgt', 'education', 'occupation', 'relationship',
       'race', 'gender', 'capital-gain', 'capital-loss', 'hours-per-week',
       'native-country', 'income'],
      dtype='object')
Satır Sayısı: 44509
```

4. Keşifsel Veri Analizi (EDA)

4.1 Yaş Dağılımı Analizi

Yaşları belirli gruplara ayırmak için **categorize_age** fonksiyonu yazıldı ve her bireyin yaşını gruba atamak için uygulandı.

Her yaş grubunun toplamındaki oranını anlamak için **value_counts(normalize=True)** kullanıldı.

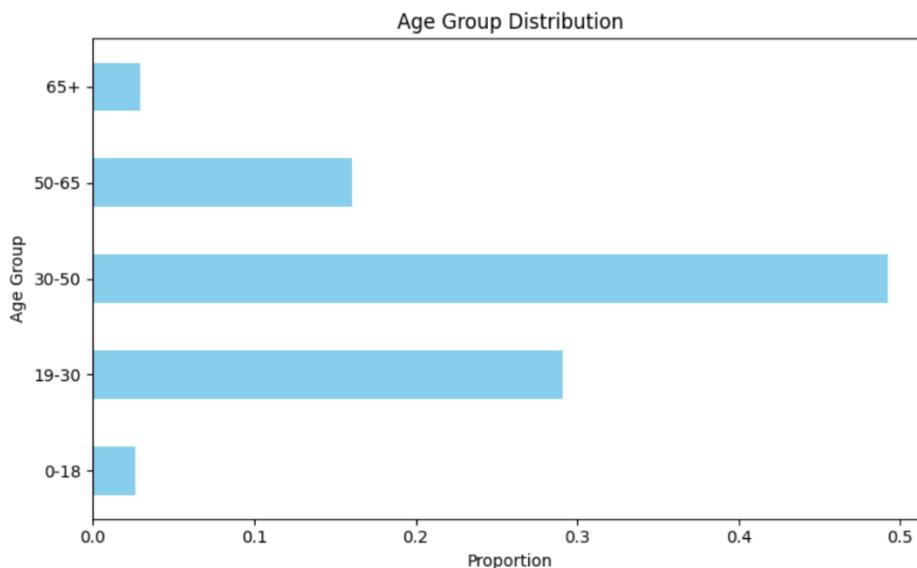
Dağılımı anlamlı bir şekilde sunmak için yatay bar grafiği çizildi.

```
# Yaşları kategorilere ayırma fonksiyonu
def categorize_age(age):
    if age <= 18:
        return '0-18'
    elif age <= 30:
        return '19-30'
    elif age <= 50:
        return '30-50'
    elif age <= 65:
        return '50-65'
    else:
        return '65+'

# Yaş gruplarını oluşturma
age_groups = df_del['age'].apply(categorize_age)

# Yaş gruplarına göre yüzdelik dağılım
age_distribution = age_groups.value_counts(normalize=True).sort_index()

# Yiyeılmış bar grafiği çizimi
age_distribution.plot(kind='barh', color='skyblue', figsize=(8, 5))
plt.xlabel('Proportion')
plt.ylabel('Age Group')
plt.title('Age Group Distribution')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

30-50 Yaş Grubu:

- En yüksek yüzdelik dilime sahiptir (~%50). Bu durum, veri setinde orta yaş grubunun ağırlıklı olarak temsil edildiğini gösterir.

19-30 Yaş Grubu:

- Yaklaşık %30'luk bir orana sahiptir ve ikinci en büyük yaş grubudur. Bu, genç yetişkinlerin veri setinde önemli bir yer tuttuğunu göstermektedir.

Düzenleme Yaşa Girenler:

- 50-65 yaş grubu:** Oranı düşüktür (~%15-20).
- 0-18 yaş ve 65+ yaş grupları:** Veri setinde oldukça az temsil edilmektedir (~%5'ten az).

Grafik, yaş gruplarına göre bireylerin dağılımını açıkça göstermektedir. En dikkat çeken bulgular şunlardır:

- 30-50 yaş grubu**, grafikte en uzun çubuk olarak öne çıkmaktadır.
- Genç yetişkinler (19-30 yaş)** ikinci sırada yer almaktadır, bu da veri setinin demografik yapısını anlamak için önemli bir ipucu sağlar.

4.1.2 Yaş Gruplarına Göre Gelir Dağılımı Analizi

Veriyi analiz edilebilir ve daha okunabilir hale getirmek için kategorilere ayırma ve etiketleme işlemleri yapıldı. (**categorize_age, map_income_label**)

Veriyi yaş gruplarına göre böldük ve gelir seviyelerinin dağılımını yüzdelik olarak hesapladık.

Yaş gruplarına göre gelir seviyelerinin yüzdelik dağılımını yiğilmiş bar grafiğiyle sunarak daha kolay anlaşılabilir bir çıktı elde ettik.

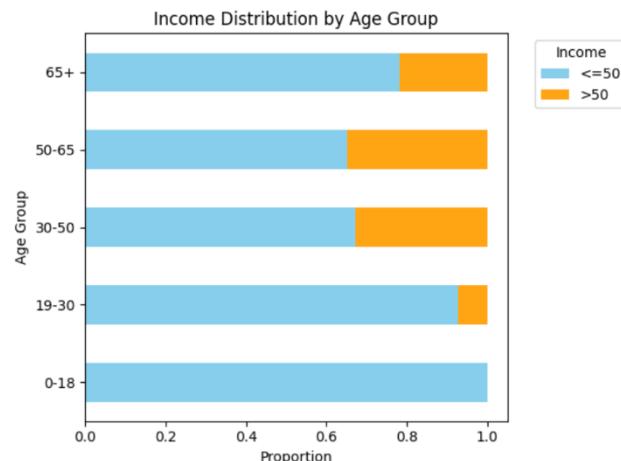
```
# Yaşları kategorilere ayırma fonksiyonu
def categorize_age(age):
    if age <= 18:
        return '0-18'
    elif age <= 30:
        return '19-30'
    elif age <= 50:
        return '30-50'
    elif age <= 65:
        return '50-65'
    else:
        return '65+'

# Income'i etiketlerle eşlestiren bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

# Yaş gruplarına ve gelir etiketlerine göre grupta
grouped_data = df_del.groupby(df_del['age'].apply(categorize_age))['income'].value_counts(normalize=True).unstack().fillna(0)

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
grouped_data.columns = grouped_data.columns.map(map_income_label)

# Yiğilmiş bar grafiği çizimi
grouped_data.plot(kind='barh', stacked=True, color=['skyblue', 'orange'])
plt.xlabel('Proportion')
plt.ylabel('Age Group')
plt.title('Income Distribution by Age Group')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

Genel Gözlemler:

- Tüm yaş gruplarında, $\leq 50K$ gelir seviyesine sahip bireylerin oranı, $>50K$ gelir seviyesine sahip bireylerden daha yüksektir.

Yaş Gruplarına Göre Gelir Dağılımı:

- 0-18 yaş grubu:** Bu grupta bireylerin tamamı $\leq 50K$ gelir seviyesine sahiptir.
- 19-30 yaş grubu:** Bu grupta $\leq 50K$ gelir seviyesi baskındır, ancak $>50K$ gelir seviyesine sahip bireyler de küçük bir orana sahiptir.
- 30-50 yaş grubu:** $>50K$ gelir seviyesine sahip bireylerin oranı bu grupta önemli ölçüde artmaktadır.
- 50-65 yaş grubu:** $>50K$ gelir seviyesindeki bireylerin oranı, bu grupta maksimum düzeye ulaşmaktadır.

- **65+ yaş grubu:** $\leq 50K$ gelir seviyesi bu grupta tekrar ağırlıklı hale gelmiştir.

Grafik, yaş gruplarına göre gelir dağılımını net bir şekilde ortaya koymaktadır:

- **$\leq 50K$ gelir seviyesi:** Tüm yaş gruplarında baskın olmasına rağmen, 30-50 ve 50-65 yaş gruplarında $> 50K$ gelir seviyesinin dikkate değer bir artış gösterdiği görülmektedir.
- **$> 50K$ gelir seviyesi:** Yaş arttıkça artış göstermeye, ancak 65+ yaş grubunda tekrar azalmaktadır.

4.2 Çalışma Sınıfı Dağılımı Analizi

Çalışma sınıflarını anlamlı hale getirmek için **workclass_mapping** ve **map()** kullanıldı.

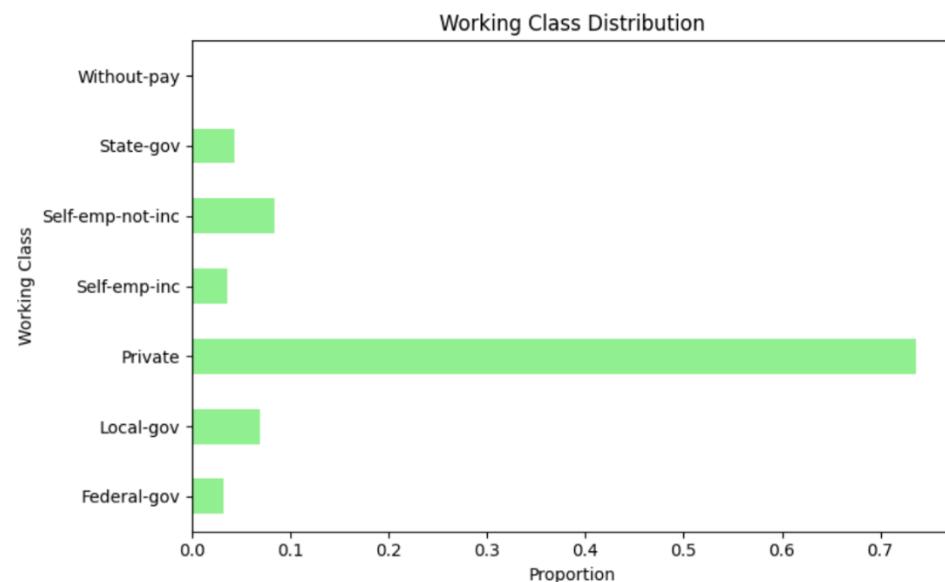
Çalışma sınıfı verilerini daha anlaşılır ve analiz edilebilir hale getirildi. Hem oranların hesaplanması hem de görselleştirilmesi yapıldı.

```
# 'workclass' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
workclass_mapping = {
    0: "Never-worked",
    1: "Without-pay",
    2: "Federal-gov",
    3: "Local-gov",
    4: "State-gov",
    5: "Self-emp-not-inc",
    6: "Self-emp-inc",
    7: "Private"
}

# Çalışma sınıflarını anlamlı isimlere çevirme
workclass_groups = df_del['workclass'].map(workclass_mapping)

# Çalışma sınıflarının yüzdelik dağılımını hesaplama
workclass_distribution = workclass_groups.value_counts(normalize=True).sort_index()

# Yıığılmış bar grafiği çizimi
workclass_distribution.plot(kind='barh', color='lightgreen', figsize=(8, 5))
plt.xlabel('Proportion')
plt.ylabel('Working Class')
plt.title('Working Class Distribution')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yüksek Çalışma Sınıfı:

- **Private** (özel sektör): Grafik, en büyük yüzdelik dilimi oluşturdu (~%70). Bu, bireylerin çoğunun özel sektörde çalıştığını gösteriyor.

Diger Çalışma Sınıfları:

- **Self-emp-not-inc** ve **Self-emp-inc** (bağımsız çalışanlar): Daha düşük oranlara sahip.
- **Local-gov, State-gov, ve Federal-gov**: Kamu sektöründe çalışan bireylerin oranı birbirine benzer ve düşüktür.
- **Without-pay** ve **Never-worked**: Çok düşük bir oranla temsil edilmiştir.

Grafik, çalışma sınıflarının dağılımını net bir şekilde göstermektedir:

- **Private** sınıfı, diğer sınıflara kıyasla açık ara en büyük grubu oluşturmaktadır.
- Kamu sektörü ve bağımsız çalışanların oranları görece daha azdır.

4.2.1 Çalışma Sınıflarına Göre Gelir Dağılımı Analizi

Çalışma sınıflarını anlamlı hale getirmek için **workclass_mapping** ve **map()** kullanıldı.

Çalışma sınıflarına göre gelir seviyelerinin yüzdelik dağılımını hesaplamak için **groupby()** ve **value_counts(normalize=True)** kullanıldı.

plot(kind='barh'): Yüzdelik dağılımı görselleştirerek analiz edilmesini sağladı.

Bu kodlar, çalışma sınıfı ve gelir seviyesi arasındaki ilişkiyi anlamak ve görselleştirmek için kullanıldı.

```
# 'workclass' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
workclass_mapping = {
    0: "Never-worked",
    1: "Without-pay",
    2: "Federal-gov",
    3: "Local-gov",
    4: "State-gov",
    5: "Self-emp-not-inc",
    6: "Self-emp-inc",
    7: "Private"
}

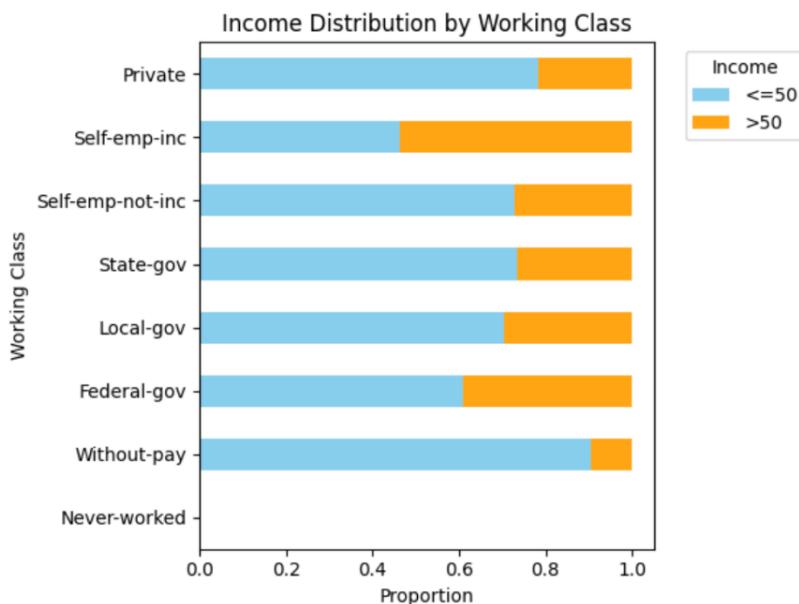
# Income'i etiketlerle eşlestiren bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

# 'workclass' sütunu anlamlı isimlerle eşleme ve yüzdelik dağılım
income_distribution = df_del.groupby(df_del['workclass'].map(workclass_mapping))['income'].value_counts(normalize=True).unstack(fill_value=0)

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
income_distribution.columns = income_distribution.columns.map(map_income_label)

# Encoding sırasına göre çalışma sınıflarını sıralama
ordered_workclasses = [workclass_mapping[i] for i in sorted(workclass_mapping.keys())]
income_distribution = income_distribution.reindex(ordered_workclasses)

# Yığılmış bar grafiği çizimi
plt.figure(figsize=(10, 6))
income_distribution.plot(kind='barh', stacked=True, color=['skyblue', 'orange'])
plt.xlabel('Proportion')
plt.ylabel('Working Class')
plt.title('Income Distribution by Working Class')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yüksek Çalışma Sınıfı:

- **Private (Özel sektör):** $\leq 50K$ gelir seviyesinde en büyük grubu oluşturdu, ancak $>50K$ gelir seviyesinde de dikkate değer bir temsil oranına sahip.

Bağımsız Çalışanlar:

- **Self-emp-inc ve Self-emp-not-inc:** $>50K$ gelir seviyesine sahip bireylerin oranı diğer gruplara göre daha yüksek.

Kamu Sektörü:

- **Federal-gov, Local-gov, ve State-gov:** Gelir seviyeleri daha dengeli dağılmış, ancak $\leq 50K$ oranı hala baskın.

Çalışmayan Gruplar:

- **Never-worked ve Without-pay:** Bu gruplar neredeyse tamamen $\leq 50K$ gelir seviyesine sahiptir.

Grafik, çalışma sınıflarına göre gelir seviyelerinin dağılımını net bir şekilde ortaya koymaktadır:

- **Private (Özel sektör)** en büyük grubu oluşturmaktır ve gelir seviyeleri arasında bir fark gözlemlenmemektedir.
- Bağımsız çalışanlar (Self-employed) diğer çalışma sınıflarına kıyasla $>50K$ gelir seviyesinde daha yüksek bir temsil oranına sahiptir.

4.3 Eğitim Seviyesi Dağılımı Analizi

Bu kodlar, 'education' sütunu için kodlanmış değerleri anlamlı isimlere çevirip yüzdelik dağılımlarını hesaplayarak, mantıksal bir sırayla sıralanmış yatay bir bar grafiği ile görselleştirmek için kullanılmıştır.

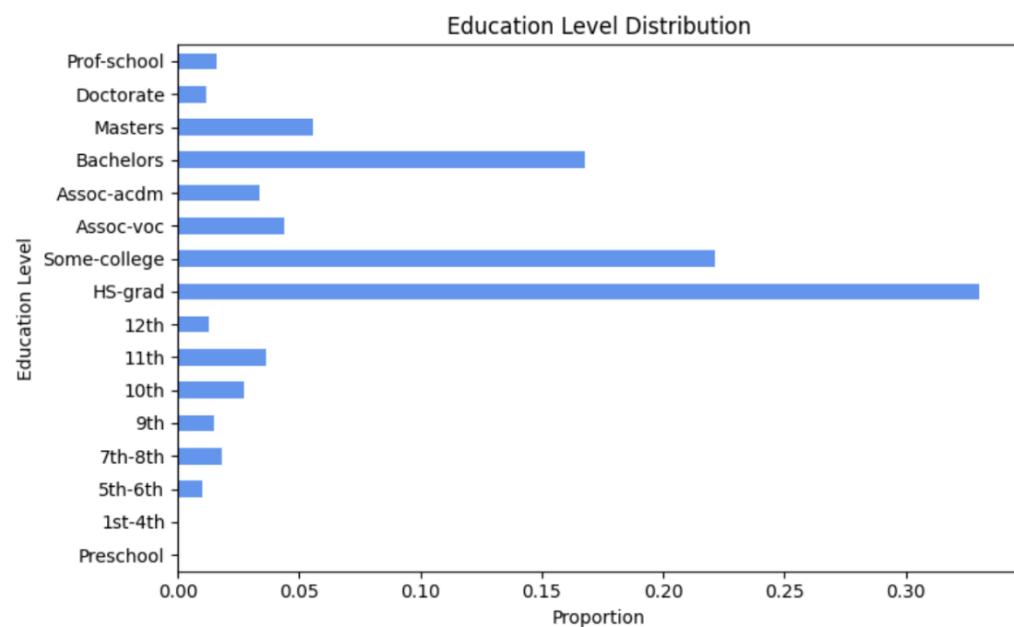
```
#'education' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
education_mapping = {
    0: "Preschool",
    1: "1st-4th",
    2: "5th-6th",
    3: "7th-8th",
    4: "9th",
    5: "10th",
    6: "11th",
    7: "12th",
    8: "HS-grad",
    9: "Some-college",
    10: "Assoc-voc",
    11: "Assoc-acdm",
    12: "Bachelors",
    13: "Masters",
    14: "Doctorate",
    15: "Prof-school"
}

# Eğitim seviyelerini anlamlı isimlere çevirme
education_groups = df_del['education'].map(education_mapping)

# Eğitim seviyelerinin yüzdelik dağılımını hesaplama
education_distribution = education_groups.value_counts(normalize=True)

# Eğitim seviyelerini sıralama (mapping sırasına göre)
ordered_education = [education_mapping[i] for i in sorted(education_mapping.keys())]
education_distribution = education_distribution.reindex(ordered_education)

# Yığılmış bar grafiği çizimi
education_distribution.plot(kind='barh', color='cornflowerblue', figsize=(8, 5))
plt.xlabel('Proportion')
plt.ylabel('Education Level')
plt.title('Education Level Distribution')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yaygın Eğitim Seviyesi:

- **HS-grad (Lise mezunu)**, en yüksek yüzdelik dilime sahiptir (~%30). Bu, veri setinde çoğu bireyin lise mezunu olduğunu göstermektedir.

Yüksek Öğrenim:

- **Some-college** ve **Bachelors** seviyeleri de yüksek oranlara sahiptir (~%20).

Düşük Eğitim Seviyeleri:

- **Preschool, 1st-4th**, ve benzeri seviyeler oldukça düşük oranlarla temsil edilmektedir.

İleri Eğitim Seviyeleri:

- **Masters, Doctorate**, ve **Prof-school** seviyelerindeki bireyler nispeten daha düşük oranlara sahiptir.

Grafik, eğitim seviyelerinin dağılımını net bir şekilde göstermektedir:

- **Lise mezunları (HS-grad)** ve bazı üniversite eğitimi alan bireyler (**Some-college**) en büyük gruppardır.
- İleri eğitim seviyelerinde (**ör. Masters, Doctorate**) bireylerin oranı oldukça düşüktür.

4.3.1 Eğitim Seviyelerine Göre Gelir Dağılımı Analizi

Bu kodlar, 'education' sütunundaki eğitim seviyelerini anlamlı isimlere çevirip, her eğitim seviyesi için gelir seviyelerinin ($\leq 50K$ ve $> 50K$) yüzdelik dağılımlarını hesaplayarak, sıralı bir yatay yığılmış bar grafiği ile görselleştirmek için kullanılmıştır.

```
# 'education' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
education_mapping = {
    0: "Preschool",
    1: "1st-4th",
    2: "5th-6th",
    3: "7th-8th",
    4: "9th",
    5: "10th",
    6: "11th",
    7: "12th",
    8: "HS-grad",
    9: "Some-college",
    10: "Assoc-voc",
    11: "Assoc-acdm",
    12: "Bachelors",
    13: "Masters",
    14: "Doctorate",
    15: "Prof-school"
}

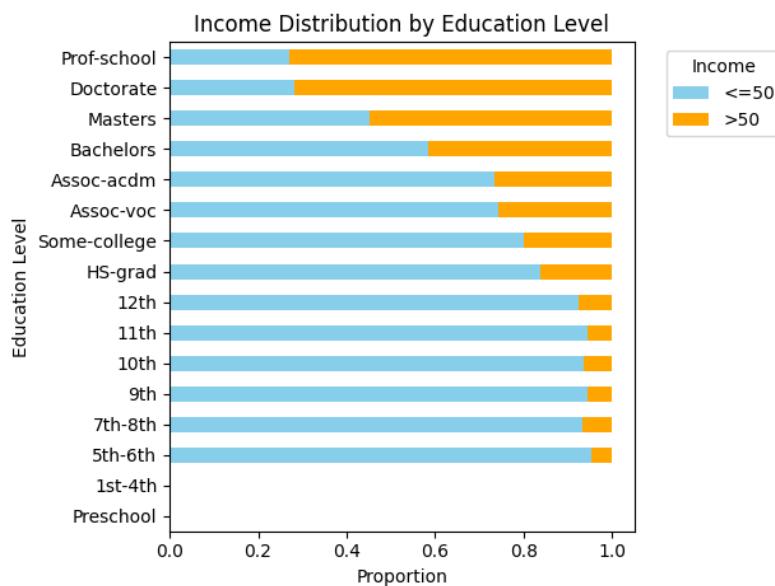
# Income'i etiketlerle eşleştirilen bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

# 'education' sütununu anlamlı isimlere eşleme ve yüzdelik dağılım
income_distribution = df_del.groupby(df_del['education'].map(education_mapping))['income'].value_counts(normalize=True).unstack()

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
income_distribution.columns = income_distribution.columns.map(map_income_label)

# Encoding sırasına göre eğitim seviyelerini sıralama
income_distribution = income_distribution.reindex([education_mapping[i] for i in sorted(education_mapping.keys())])

# Yığılmış bar grafiği çizimi
plt.figure(figsize=(10, 6))
income_distribution.plot(kind='barh', stacked=True, color=['skyblue', 'orange'])
plt.xlabel('Proportion')
plt.ylabel('Education Level')
plt.title('Income Distribution by Education Level')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yüksek Gelir Seviyesi:

- **Prof-school, Doctorate, ve Masters** seviyelerinde >50K gelir seviyesine sahip bireylerin oranı diğer seviyelere göre daha yüksektir.

Düşük Gelir Seviyeleri:

- **Preschool** ve **1st-4th** gibi düşük eğitim seviyelerinde bireylerin neredeyse tamamı ≤50K gelir seviyesine sahiptir.

Lise ve Üniversite Seviyeleri:

- **HS-grad** ve **Some-college** seviyelerinde ≤50K gelir oranı baskındır, ancak >50K geliri olan bireylerin oranı da gözlemlenebilir.

Grafik, eğitim seviyelerine göre gelir seviyelerinin yüzdelik dağılımını net bir şekilde ortaya koymaktadır:

- Daha yüksek eğitim seviyeleri, >50K gelir seviyesinin artışıyla ilişkilidir.
- Düşük eğitim seviyeleri, genellikle ≤50K gelir seviyesinin baskın olduğunu göstermektedir.

4.4 İlişki Durumu Dağılımı Analizi

Bu kodlar, 'relationship' sütunundaki ilişki durumlarını anlamlı isimlere çevirip, her ilişki durumu için yüzdelik dağılımı hesaplayarak, sıralı bir yatay bar grafiği ile görselleştirmek için kullanılmıştır.

```

# 'relationship' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
relationship_mapping = {
    0: "Other-relative",
    1: "Not-in-family",
    2: "Own-child",
    3: "Unmarried",
    4: "Wife",
    5: "Husband"
}

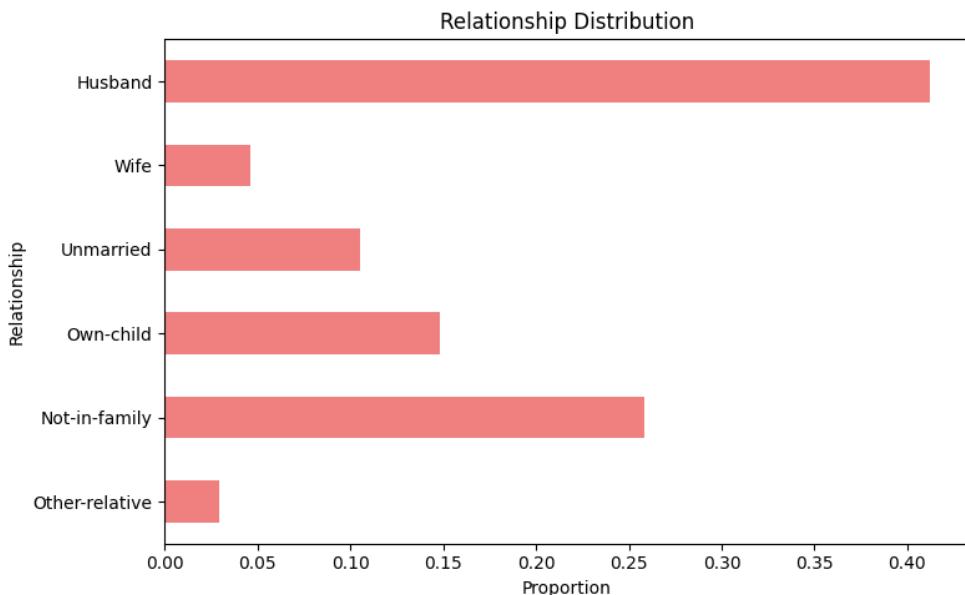
# Relationship kategorilerini anlamlı isimlere çevirme
relationship_groups = df_del['relationship'].map(relationship_mapping)

# Relationship seviyelerinin yüzdelik dağılımını hesaplama
relationship_distribution = relationship_groups.value_counts(normalize=True)

# Relationship seviyelerini sıralama (mapping sırasına göre)
ordered_relationships = [relationship_mapping[i] for i in sorted(relationship_mapping.keys())]
relationship_distribution = relationship_distribution.reindex(ordered_relationships)

# Yiğilmiş bar grafiği çizimi
relationship_distribution.plot(kind='barh', color='lightcoral', figsize=(8, 5))
plt.xlabel('Proportion')
plt.ylabel('Relationship')
plt.title('Relationship Distribution')
plt.tight_layout()
plt.show()

```



ELDE EDİLEN SONUÇLAR:

En Yaygın İlişki Durumu:

- Husband** (Koca): Veri setindeki bireylerin en büyük yüzdesini oluşturmaktadır (~%40).

Düzen İlişki Durumları:

- Not-in-family** (Ailede değil): İkinci büyük yüzdelik dilimi (~%25).
- Own-child** ve **Unmarried**: Daha düşük oranlarla temsil edilmiştir.
- Wife** (Eş) ve **Other-relative**: En az temsil edilen ilişki durumlarıdır.

Grafik, ilişki durumlarının yüzdelik dağılımını açıkça göstermektedir:

- Husband** ve **Not-in-family**, en baskın ilişki durumlarıdır.
- Other-relative** ve **Wife**, daha az temsil edilen ilişki gruplarıdır.

4.4.1 İlişki Durumlarına Göre Gelir Dağılımı Analizi

Bu kodlar, 'relationship' sütunundaki ilişki durumlarını anlamlı isimlere çevirip, her ilişki durumu için gelir seviyelerinin ($\leq 50K$ ve $>50K$) yüzdelik dağılımlarını hesaplayarak, sıralı bir yatay yığılmış bar grafiği ile görselleştirmek için kullanılmıştır.

```
# 'relationship' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
relationship_mapping = {
    0: "Other-relative",
    1: "Not-in-family",
    2: "Own-child",
    3: "Unmarried",
    4: "Wife",
    5: "Husband"
}

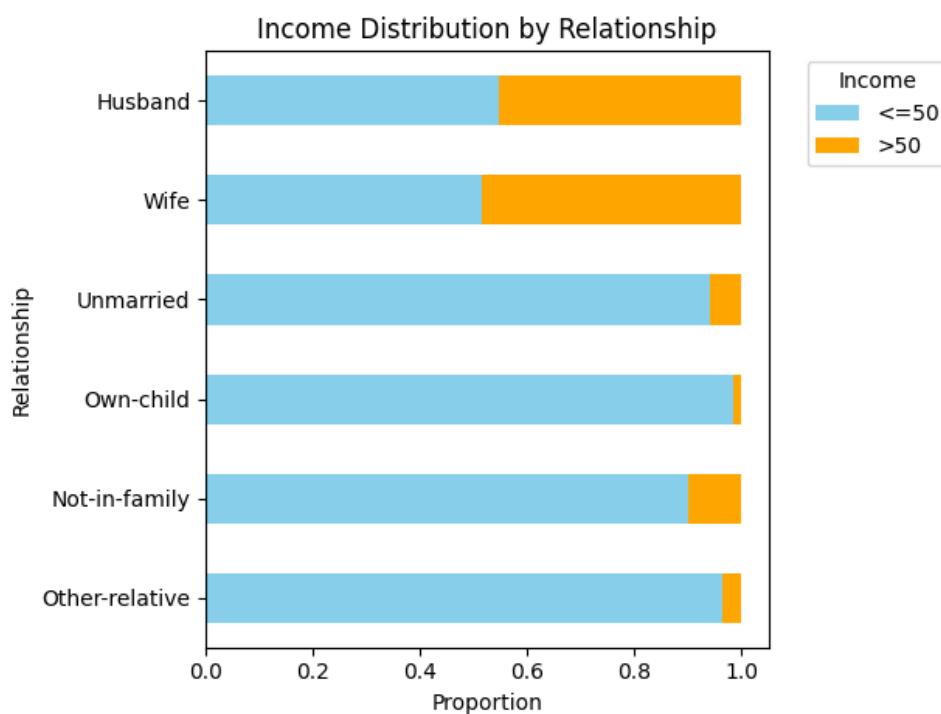
# Income'i etiketlerle eşleştiren bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

# 'relationship' sütununu anlamlı isimlere eşleme ve yüzdelik dağılım
income_distribution = df_del.groupby(df_del['relationship'].map(relationship_mapping))['income'].value_counts(normalize=True).unstack()

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
income_distribution.columns = income_distribution.columns.map(map_income_label)

# Mertebe sırasına göre yeniden düzenleme
ordered_relationships = [relationship_mapping[i] for i in sorted(relationship_mapping.keys())]
income_distribution = income_distribution.reindex(ordered_relationships)

# Yığılmış bar grafiği çizimi
plt.figure(figsize=(10, 6))
income_distribution.plot(kind='barh', stacked=True, color=['skyblue', 'orange'])
plt.xlabel('Proportion')
plt.ylabel('Relationship')
plt.title('Income Distribution by Relationship')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yüksek Gelir Seviyesi:

- **Husband (Koca)** ve **Wife (Eş)** gruplarında $>50K$ gelir seviyesine sahip bireylerin oranı diğer gruplara göre daha yüksektir.

Düşük Gelir Seviyesi:

- **Own-child, Not-in-family**, ve **Other-relative** gruplarında bireylerin büyük çoğunluğu $\leq 50K$ gelir seviyesine sahiptir.

Unmarried ve Not-in-family:

- Bu gruplar arasında $>50K$ gelir seviyesine sahip bireylerin oranı oldukça düşüktür.

Grafik, ilişki durumlarına göre gelir seviyelerinin dağılımını net bir şekilde göstermektedir:

- **Husband** grubu en büyük $>50K$ oranına sahiptir.
- **Other-relative** ve **Own-child** gibi gruplar neredeyse tamamen $\leq 50K$ seviyesinde temsil edilmektedir.

4.5 Meslek Dağılımı Analizi

Bu kodlar, 'occupation' sütunundaki meslek gruplarını anlamlı isimlere çevirip, her meslek grubunun yüzdelik dağılımını hesaplayarak, sıralı bir yatay bar grafiği ile görselleştirmek için kullanılmıştır.

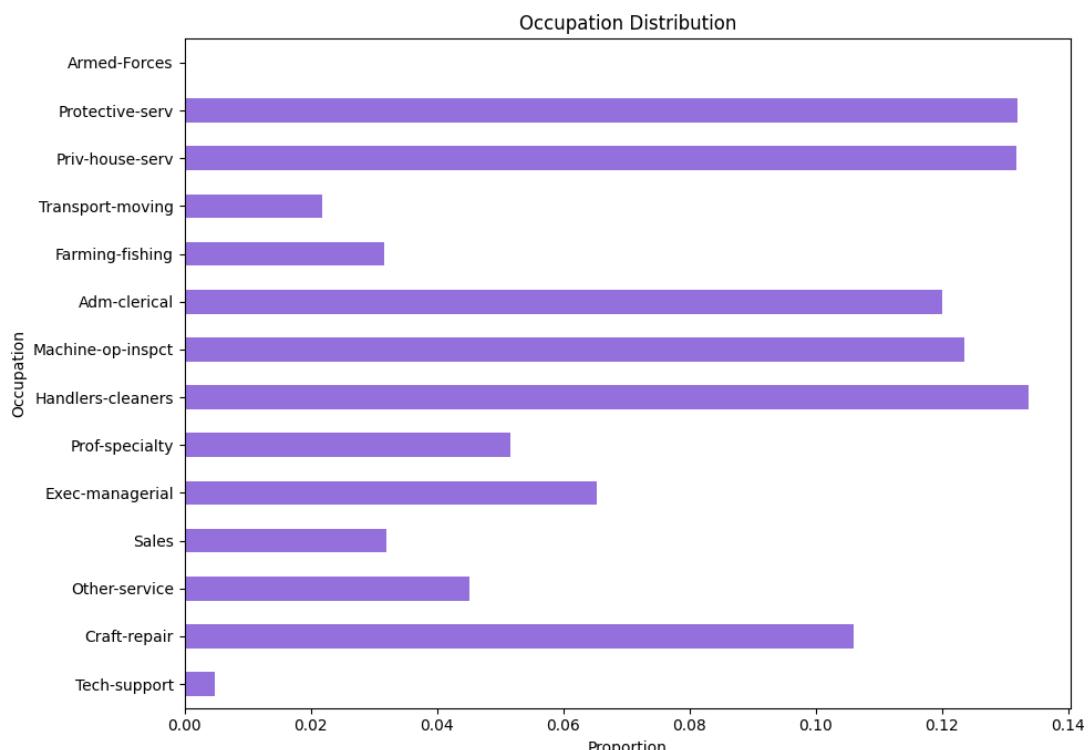
```
# 'occupation' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
occupation_mapping = {
    0: "Tech-support",
    1: "Craft-repair",
    2: "Other-service",
    3: "Sales",
    4: "Exec-managerial",
    5: "Prof-specialty",
    6: "Handlers-cleaners",
    7: "Machine-op-inspct",
    8: "Adm-clerical",
    9: "Farming-fishing",
    10: "Transport-moving",
    11: "Priv-house-serv",
    12: "Protective-serv",
    13: "Armed-Forces"
}

# Occupation kategorilerini anlamlı isimlere çevirme
occupation_groups = df['occupation'].map(occupation_mapping)

# Occupation seviyelerinin yüzdelik dağılımını hesaplama
occupation_distribution = occupation_groups.value_counts(normalize=True)

# Occupation seviyelerini sıralama (mapping sırasına göre)
ordered_occupations = [occupation_mapping[i] for i in sorted(occupation_mapping.keys())]
occupation_distribution = occupation_distribution.reindex(ordered_occupations)

# Yığılmış bar grafiği çizimi
occupation_distribution.plot(kind='barh', color='mediumpurple', figsize=(10, 7))
plt.xlabel('Proportion')
plt.ylabel('Occupation')
plt.title('Occupation Distribution')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yaygın Meslek Grupları:

- **Handlers-cleaners**, **Adm-clerical**, ve **Exec-managerial** grupları en yüksek yüzdelik dilime sahiptir.

Daha Az Temsil Edilen Meslekler:

- **Armed-Forces**, **Protective-serv**, ve **Tech-support** grupları en düşük oranlarla temsil edilmektedir.

Grafik, mesleklerin veri setindeki dağılımını açıkça göstermektedir:

- Bazı meslek grupları (örneğin, **Handlers-cleaners** ve **Adm-clerical**) diğer gruplara göre daha yaygındır.
- **Armed-Forces** gibi gruplar ise nadiren temsil edilmiştir.

4.5.1 Meslekler Göre Gelir Dağılımı Analizi

Bu kodlar, 'occupation' sütunundaki meslek gruplarını anlamlı isimlere çevirip, her meslek grubu için gelir seviyelerinin ($\leq 50K$ ve $>50K$) yüzdelik dağılımlarını hesaplayarak, sıralı bir yatay yığılmış bar grafiği ile görselleştirmek için kullanılmıştır.

```

# 'occupation' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
occupation_mapping = {
    0: "Priv-house-serv",
    1: "Other-service",
    2: "Handlers-cleaners",
    3: "Farming-fishing",
    4: "Machine-op-inspct",
    5: "Transport-moving",
    6: "Craft-repair",
    7: "Adm-clerical",
    8: "Sales",
    9: "Tech-support",
    10: "Protective-serv",
    11: "Prof-specialty",
    12: "Exec-managerial",
    13: "Armed-Forces"
}

# Income'i etiketlerle eşleştirilen bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

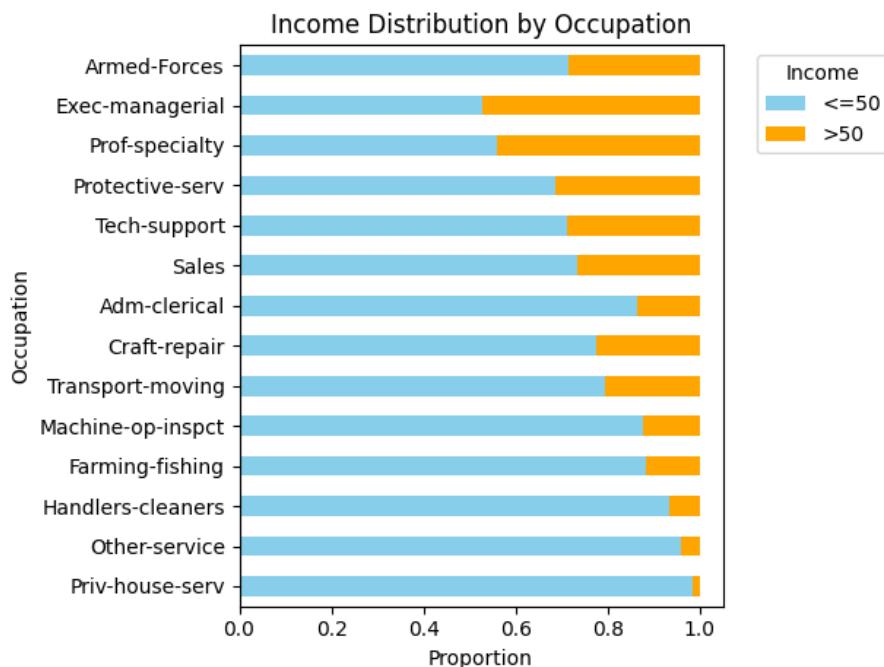
# 'occupation' sütununu anlamlı isimlerle eşleme ve yüzdelik dağılım
income_distribution = df_del.groupby(df_del['occupation'].map(occupation_mapping))['income'].value_counts(normalize=True).unstack()

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
income_distribution.columns = income_distribution.columns.map(map_income_label)

# Encoding sırasına göre meslekleri sıralama
ordered_occupations = [occupation_mapping[i] for i in sorted(occupation_mapping.keys())]
income_distribution = income_distribution.reindex(ordered_occupations)

# Yiğilmiş bar grafiği çizimi
plt.figure(figsize=(10, 6))
income_distribution.plot(kind='barh', stacked=True, color=['skyblue', 'orange'])
plt.xlabel('Proportion')
plt.ylabel('Occupation')
plt.title('Income Distribution by Occupation')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

```



ELDE EDİLEN SONUÇLAR:

En Yüksek Gelir Seviyesi:

- **Exec-managerial** ve **Prof-specialty** gruplarında >50K gelir seviyesine sahip bireylerin oranı diğer gruplara göre daha yüksektir.

Düşük Gelir Seviyeleri:

- **Priv-house-serv**, **Handlers-cleaners**, ve **Other-service** gruplarında bireylerin büyük çoğunluğu $\leq 50K$ gelir seviyesine sahiptir.

Meslekler Arası Gelir Farklılıkları:

- Teknik ve yönetim meslek gruplarında $> 50K$ oranı artış gösterirken, hizmet sektöründe $\leq 50K$ oranı baskındır.

Grafik, mesleklerin gelir seviyeleri üzerindeki etkisini net bir şekilde göstermektedir:

- Yüksek gelir seviyeleri genellikle yönetim ve uzmanlık gerektiren mesleklerde yaygındır.
- Düşük gelir seviyeleri daha çok hizmet sektörüne ait mesleklerde görülmektedir.

4.6 İrk Dağılımı Analizi

Bu kodlar, 'race' sütunundaki ırk gruplarını anlamlı isimlere çevirip, her ırk grubunun yüzdelik dağılımını hesaplayarak, sıralı bir yatay bar grafiği ile görselleştirmek için kullanılmıştır.

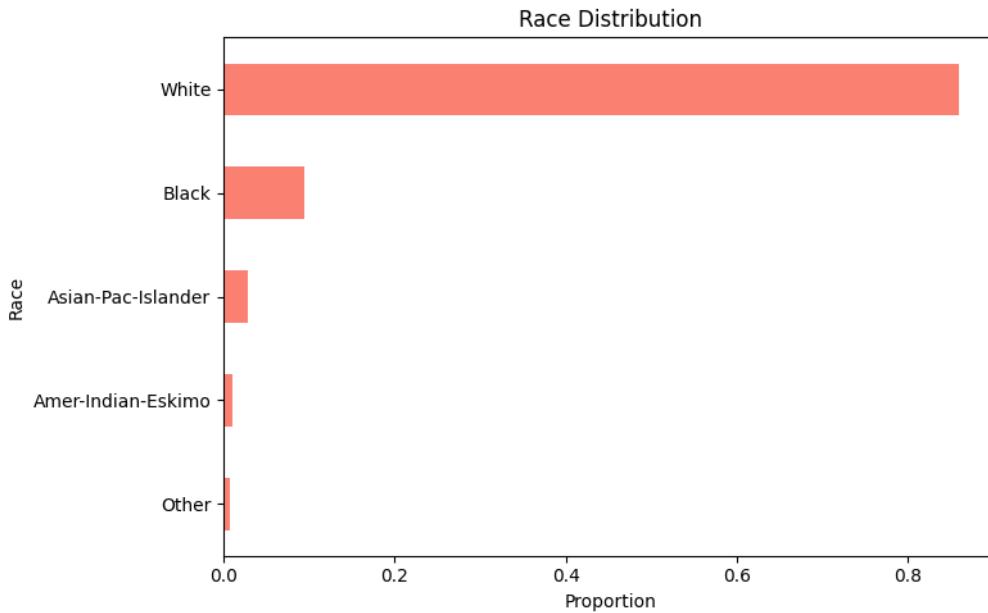
```
# 'race' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
race_mapping = {
    0: "Other",
    1: "Amer-Indian-Eskimo",
    2: "Asian-Pac-Islander",
    3: "Black",
    4: "White"
}

# İrk kategorilerini anlamlı isimlere çevirme
race_groups = df_del['race'].map(race_mapping)

# İrk seviyelerinin yüzdelik dağılımını hesaplama
race_distribution = race_groups.value_counts(normalize=True)

# İrk seviyelerini sıralama (mapping sırasına göre)
ordered_races = [race_mapping[i] for i in sorted(race_mapping.keys())]
race_distribution = race_distribution.reindex(ordered_races)

# Yiğilmiş bar grafiği çizimi
race_distribution.plot(kind='barh', color='salmon', figsize=(8, 5))
plt.xlabel('Proportion')
plt.ylabel('Race')
plt.title('Race Distribution')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yaygın İrk Grubu:

- **White (Beyaz)**: Veri setinde baskın ırk grubudur (~%80).

Daha Az Temsil Edilen İrkler:

- **Black, Asian-Pac-Islander, ve Amer-Indian-Eskimo** grupları daha az temsil edilmektedir.
- **Other (Diğer)** grubu en düşük yüzdelik dilime sahiptir.

Grafik, veri setinde farklı ırk gruplarının temsil oranlarını açıkça göstermektedir:

- **White** grubu, açık ara en yüksek orana sahiptir.
- Diğer ırk grupları oldukça düşük oranlarda temsil edilmiştir.

4.6.1 İrlklara Göre Gelir Dağılımı Analizi

Bu kodlar, 'race' sütunundaki ırk gruplarını anlamlı isimlere çevirip, her ırk grubu için gelir seviyelerinin ($\leq 50K$ ve $>50K$) yüzdelik dağılımlarını hesaplayarak, sıralı bir yatay yiğilmiş bar grafiği ile görselleştirmek için kullanılmıştır.

```

# 'race' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
race_mapping = {
    0: "Other",
    1: "Amer-Indian-Eskimo",
    2: "Asian-Pac-Islander",
    3: "Black",
    4: "White"
}

# Income'i etiketlerle eşlestiren bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

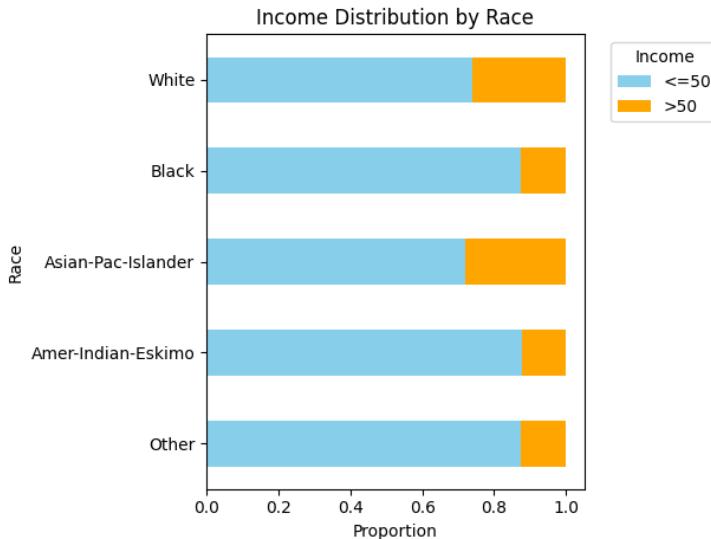
# 'race' sütununu anlamlı isimlere eşleme ve yüzdelik dağılım
income_distribution = df_del.groupby(df_del['race'].map(race_mapping))['income'].value_counts(normalize=True).unstack()

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
income_distribution.columns = income_distribution.columns.map(map_income_label)

# Encoding sırasına göre ırkları sıralama
ordered_races = [race_mapping[i] for i in sorted(race_mapping.keys())]
income_distribution = income_distribution.reindex(ordered_races)

# Yığılmış bar grafiği çizimi
plt.figure(figsize=(10, 6))
income_distribution.plot(kind='barh', stacked=True, color=['skyblue', 'orange'])
plt.xlabel('Proportion')
plt.ylabel('Race')
plt.title('Income Distribution by Race')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

```



ELDE EDİLEN SONUÇLAR:

En Yüksek Gelir Seviyesi:

Asian-Pac-Islander ve **White** gruplarında >50K gelir seviyesine sahip bireylerin oranı diğer gruplara göre daha yüksektir.

Düşük Gelir Seviyesi:

Other, **Amer-Indian-Eskimo**, ve **Black** gruplarında bireylerin büyük çoğunluğu ≤50K gelir seviyesine sahiptir.

Irklar Arası Gelir Farklılıkları:

Irklar arasında gelir farklılıklarını belirtmektedir; bazı gruplar yüksek gelir seviyelerine daha fazla sahiptir.

Grafik, farklı ırkların gelir seviyelerindeki dağılımı açıkça göstermektedir:

- **Asian-Pac-Islander** ve **White** gruplarında yüksek gelir seviyeleri daha yaygındır.
- Diğer gruplarda düşük gelir seviyeleri baskındır.

4.7 Cinsiyet Dağılımı Analizi

Bu kodlar, 'gender' sütunundaki cinsiyet gruplarını anlamlı isimlere çevirip, her cinsiyet grubunun yüzdelik dağılımını hesaplayarak, sıralı bir yatay bar grafiği ile görselleştirmek için kullanılmıştır.

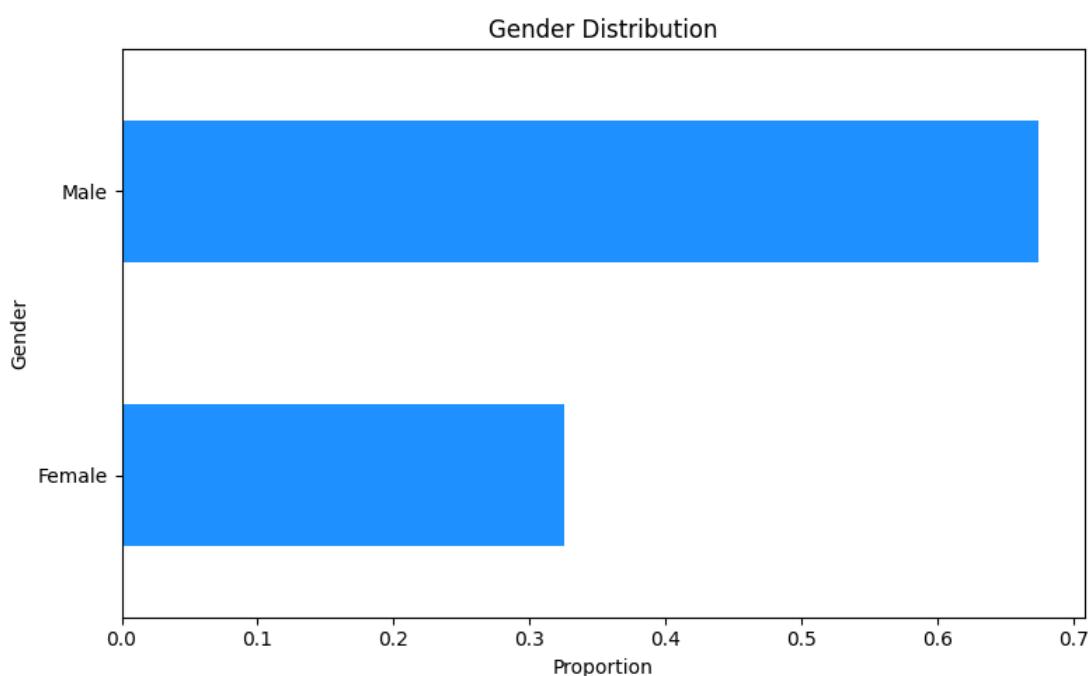
```
# 'gender' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
gender_mapping = {
    0: "Female",
    1: "Male"
}

# Gender kategorilerini anlamlı isimlere çevirme
gender_groups = df_del['gender'].map(gender_mapping)

# Gender seviyelerinin yüzdelik dağılımını hesaplama
gender_distribution = gender_groups.value_counts(normalize=True)

# Gender seviyelerini sıralama (mapping sırasına göre)
ordered_genders = [gender_mapping[i] for i in sorted(gender_mapping.keys())]
gender_distribution = gender_distribution.reindex(ordered_genders)

# Yığılmış bar grafiği çizimi
gender_distribution.plot(kind='barh', color='dodgerblue', figsize=(8, 5))
plt.xlabel('Proportion')
plt.ylabel('Gender')
plt.title('Gender Distribution')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yaygın Cinsiyet Grubu:

- **Male (Erkek)**, veri setinde baskın cinsiyet grubudur (~%70).

Daha Az Temsil Edilen Cinsiyet:

- **Female (Kadın)** grubu daha az oranla temsil edilmektedir (~%30).

Grafik, veri setinde cinsiyetlerin temsil oranlarını açıkça göstermektedir:

- **Male** grubu, açık ara en yüksek orana sahiptir.
- **Female** grubu ise daha düşük bir oranda temsil edilmiştir.

4.7.1 Cinsiyete Göre Gelir Dağılımı Analizi

Bu kodlar, 'gender' sütunundaki cinsiyet gruplarını anlamlı isimlere çevirecek, her cinsiyet grubu için gelir seviyelerinin ($\leq 50K$ ve $>50K$) yüzdelik dağılımlarını hesaplayarak, sıralı bir yatay yığılmış bar grafiği ile görselleştirmek için kullanılmıştır.

```
# 'gender' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
gender_mapping = {
    0: "Female",
    1: "Male"
}

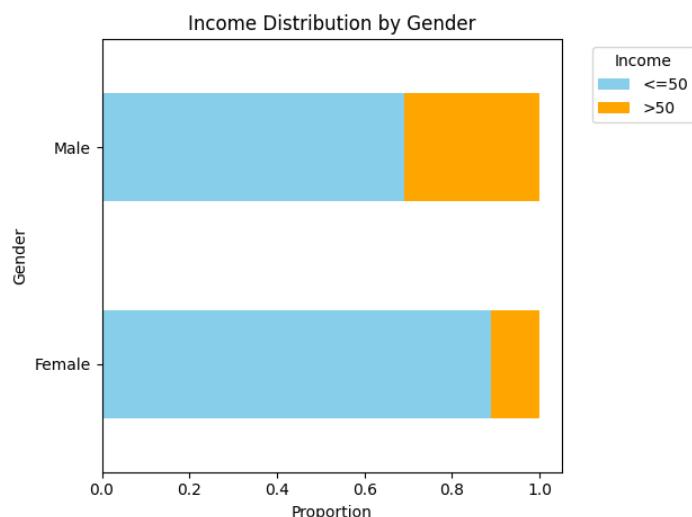
# Income'i etiketlerle eşlestiren bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

# 'gender' sütununu anlamlı isimlerle eşleme ve yüzdelik dağılımını hesapla
income_distribution = df_del.groupby(df_del['gender'].map(gender_mapping))['income'].value_counts(normalize=True).unstack()

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
income_distribution.columns = income_distribution.columns.map(map_income_label)

# Encoding sırasına göre cinsiyetleri sıralama
ordered_genders = [gender_mapping[i] for i in sorted(gender_mapping.keys())]
income_distribution = income_distribution.reindex(ordered_genders)

# Yığılmış bar grafiği çizimi
plt.figure(figsize=(8, 5))
income_distribution.plot(kind='barh', stacked=True, color=['skyblue', 'orange'])
plt.xlabel('Proportion')
plt.ylabel('Gender')
plt.title('Income Distribution by Gender')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yüksek Gelir Seviyesi:

- **Male (Erkek)** grubu, >50K gelir seviyesine sahip bireylerin oranı açısından daha yüksek bir temsil oranına sahiptir.

Düşük Gelir Seviyesi:

- **Female (Kadın)** grubunda bireylerin büyük çoğunluğu ≤50K gelir seviyesine sahiptir.

Grafik, cinsiyetlere göre gelir seviyelerindeki dağılımı net bir şekilde göstermektedir:

- **Male** grubu, >50K gelir seviyesine daha fazla sahiptir.
- **Female** grubu ise genellikle ≤50K gelir seviyesinde baskındır.

4.8 Memlekete Göre Dağılım Analizi

Bu kodlar, 'native-country' sütunundaki doğum ülkesi gruplarını anlamlı isimlere çevirip, her ülke için yüzdelik dağılımı hesaplayarak, sıralı bir yatay bar grafiği ile görselleştirmek için kullanılmıştır.

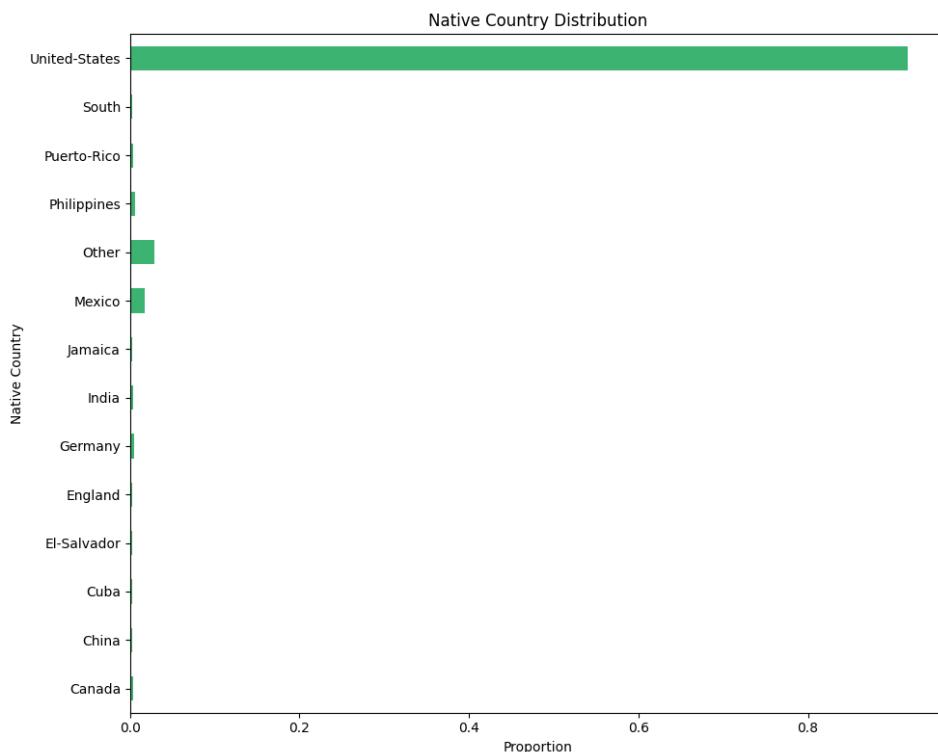
```
# 'native-country' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
native_country_mapping = {
    0: "Canada",
    1: "China",
    2: "Cuba",
    3: "El-Salvador",
    4: "England",
    5: "Germany",
    6: "India",
    7: "Jamaica",
    8: "Mexico",
    9: "Other",
    10: "Philippines",
    11: "Puerto-Rico",
    12: "South",
    13: "United-States"
}

# Native Country kategorilerini anlamlı isimlere çevirme
native_country_groups = df['native-country'].map(native_country_mapping)

# Native Country seviyelerinin yüzdelik dağılımını hesaplama
native_country_distribution = native_country_groups.value_counts(normalize=True)

# Native Country seviyelerini sıralama (mapping sırasına göre)
ordered_countries = [native_country_mapping[i] for i in sorted(native_country_mapping.keys())]
native_country_distribution = native_country_distribution.reindex(ordered_countries)

# Yığılmış bar grafiği çizimi
native_country_distribution.plot(kind='barh', color='mediumseagreen', figsize=(10, 8))
plt.xlabel('Proportion')
plt.ylabel('Native Country')
plt.title('Native Country Distribution')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yaygın Memleket:

- **United States**, veri setindeki bireylerin büyük bir kısmını oluşturmaktadır (~%90).

Daha Az Temsil Edilen Doğum Ülkeleri:

- **Mexico**, **Philippines**, ve **India** gibi ülkeler düşük oranlarla temsil edilmektedir.
- **Other** kategorisi, birçok küçük grubu içerir ve düşük bir yüzdelik dilime sahiptir.

Grafik, bireylerin memleket dağılımını açıkça göstermektedir:

- **United States**, açık ara en yüksek orana sahiptir.
- Diğer ülkeler oldukça düşük oranlarda temsil edilmiştir.

4.8.1 Memlekete Göre Gelir Dağılımı Analizi

Bu kodlar, 'native-country' sütunundaki memleket gruplarını anlamlı isimlere çevirip, her memleket için gelir seviyelerinin ($\leq 50K$ ve $> 50K$) yüzdelik dağılımlarını hesaplayarak, sıralı bir yatay yığılmış bar grafiği ile görselleştirmek için kullanılmıştır.

```

# 'native-country' sütunu için kodlanmış değerleri anlamlı isimlere çeviren eşleme
native_country_mapping = {
    0: "Canada",
    1: "China",
    2: "Cuba",
    3: "El-Salvador",
    4: "England",
    5: "Germany",
    6: "India",
    7: "Jamaica",
    8: "Mexico",
    9: "Other",
    10: "Philippines",
    11: "Puerto-Rico",
    12: "South",
    13: "United-States"
}

# Income'i etiketlerle eşlestiren bir fonksiyon
def map_income_label(value):
    return '<=50' if value == 0 else '>50'

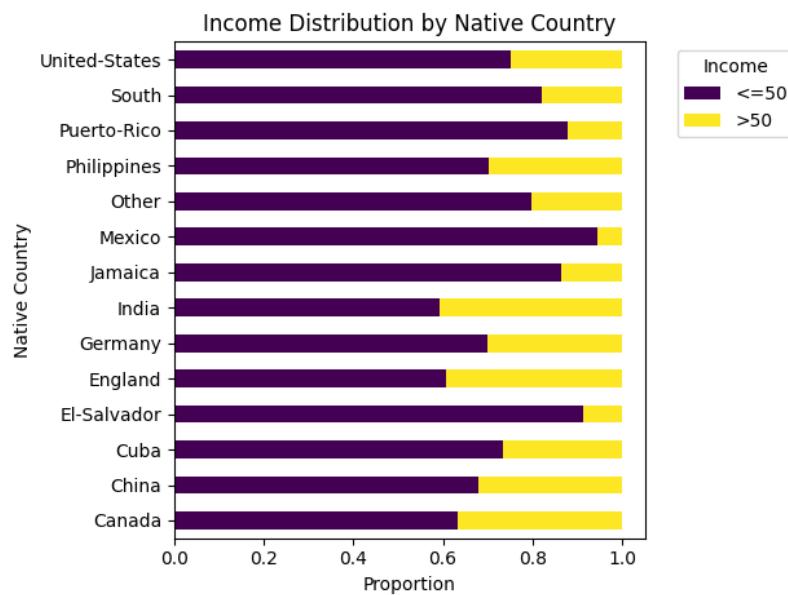
# 'native-country' sütununu anlamlı isimlere eşleme ve yüzdelik dağılım
income_distribution = df_del.groupby(df_del['native-country'].map(native_country_mapping))['income'].value_counts(normalize=True).unstack()

# Income etiketlerini değiştirme (yalnızca görselleştirme için)
income_distribution.columns = income_distribution.columns.map(map_income_label)

# Encoding sırasına göre native country kategorilerini sıralama
ordered_countries = [native_country_mapping[i] for i in sorted(native_country_mapping.keys())]
income_distribution = income_distribution.reindex(ordered_countries)

# Yığılmış bar grafiği çizimi (Y eksen: Native Country, X eksen: Proportion)
plt.figure(figsize=(12, 8))
income_distribution.plot(kind='barh', stacked=True, colormap='viridis')
plt.xlabel('Proportion')
plt.ylabel('Native Country')
plt.title('Income Distribution by Native Country')
plt.legend(title='Income', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

```



ELDE EDİLEN SONUÇLAR:

En Yüksek Gelir Seviyesi:

- **United States**, hem ≤50K hem de >50K gelir seviyelerinde en çok temsil edilen ülke grubudur.

Düşük Temsil Oranına Sahip Ülkeler:

- **Canada, China, ve Jamaica** gibi ülkeler daha az temsil edilmiştir.

Ülkeler Arası Gelir Farklılıkları:

- Bazı ülkelerde (ör. **United States**) >50K gelir seviyesinin oranı diğer ülkelere göre daha yüksektir.

Grafik, memleketlere göre gelir seviyelerindeki dağılımı açıkça göstermektedir:

- United States**, en baskın ülke grubu olarak hem düşük hem de yüksek gelir seviyelerini kapsar.
- Diğer ülkeler genellikle düşük temsil oranına sahiptir.

4.9 Normalleştirilmiş Haftalık Çalışma Saatleri Dağılımı Analizi

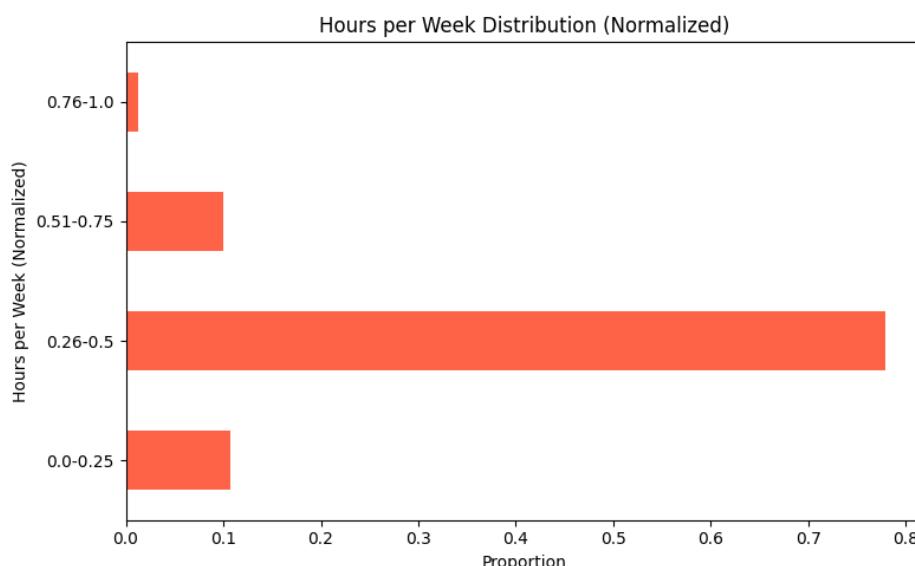
Bu kodlar, 'hours-per-week' sütunundaki çalışma saatlerini kategorilere ayırip, her kategori için yüzdelik dağılımı hesaplayarak, sıralı bir yatay bar grafiği ile görselleştirmek için kullanılmıştır.

```
# Normalleştirilmiş çalışma saatlerini kategorilere ayırma fonksiyonu
def categorize_normalized_hours(hours):
    if hours <= 0.25:
        return "0.0-0.25"
    elif hours <= 0.5:
        return "0.26-0.5"
    elif hours <= 0.75:
        return "0.51-0.75"
    else:
        return "0.76-1.0"

# Normalleştirilmiş çalışma saatlerini kategorilere ayırma
normalized_hours_groups = df_del['hours-per-week'].apply(categorize_normalized_hours)

# Çalışma saatleri kategorilerinin yüzdelik dağılımını hesaplama
normalized_hours_distribution = normalized_hours_groups.value_counts(normalize=True).sort_index()

# Bar grafiği çizimi
normalized_hours_distribution.plot(kind='barh', color='tomato', figsize=(8, 5))
plt.xlabel('Proportion')
plt.ylabel('Hours per Week (Normalized)')
plt.title('Hours per Week Distribution (Normalized)')
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

En Yaygın Çalışma Saatleri Aralığı (0.26-0.5):

- Çoğu birey bu aralıkta, yani haftalık **26-50 saat** çalışmaktadır (%70 civarı).
- Bu durum, tam zamanlı çalışan bireylerin veri setinde baskın olduğunu göstermektedir.

Düşük Çalışma Saatleri Aralığı (0.0-0.25):

- Haftalık **0-25 saat** çalışan bireyler genellikle yarı zamanlı çalışanları temsil eder ve oranları düşüktür (~%10).

Yüksek Çalışma Saatleri Aralığı (0.76-1.0):

- Haftalık **75 saat ve üzeri** çalışan bireyler oldukça azdır (~%2).
- Bu, aşırı uzun saatler çalışan bireylerin nadir olduğunu gösterir.

Orta-Yüksek Çalışma Saatleri Aralığı (0.51-0.75):

- Haftalık **51-75 saat** çalışan bireyler, orta düzey bir orana sahiptir (~%15-20).

Çıkarımlar:

- Çoğunluk tam zamanlı çalışanlardan oluşmaktadır.
- Hem yarı zamanlı çalışanlar hem de uzun süre çalışan bireyler az temsil edilmektedir.

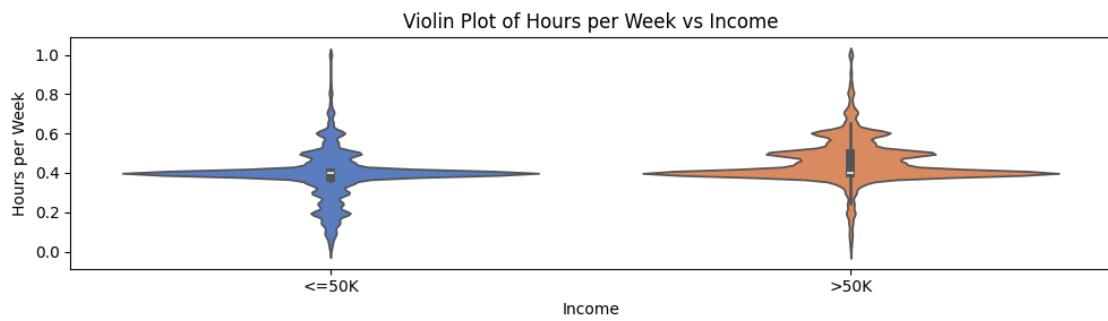
Grafik, bireylerin haftalık normalleştirilmiş çalışma saatleri dağılımını açıkça göstermektedir.

Normalleştirilmiş çalışma saatleri, veri setinin büyük kısmının tam zamanlı çalışanlardan olduğunu, azınlıkta kalan grupların ise yarı zamanlı veya uzun saatler çalışan bireylerden olduğunu ortaya koymaktadır.

4.9.1 Haftalık Çalışma Saatleri ve Gelir Dağılımı Analizi (Violin Plot)

Bu kodlar, haftalık çalışma saatlerinin gelir seviyelerine göre dağılımını violin plot ile görselleştirerek, iki grup arasındaki farklılıklarını ve benzerlikleri incelemek için kullanılmıştır.

```
# hours-per-week Vs income
plt.figure(figsize=(10, 3))
sns.violinplot(x='income', y='hours-per-week', data=df_del, palette='muted')
plt.xlabel('Income')
plt.ylabel('Hours per Week')
plt.title('Violin Plot of Hours per Week vs Income')
plt.xticks([0, 1], ['<=50K', '>50K'])
plt.tight_layout()
plt.show()
```



ELDE EDİLEN SONUÇLAR:

$\leq 50K$ Gelir Grubu:

- Çoğunluk haftalık **35-45 saat** çalışmaktadır.
- Daha geniş bir varyasyona sahip, yani düşük saatlerde çalışan bireyler (part-time) bu grupta daha yaygın.

$>50K$ Gelir Grubu:

- Haftalık çalışma saatleri genellikle **40-50 saat** arasında yoğunlaşmıştır.
- Bu gruptaki bireylerin çalışma saatleri daha odaklıdır, part-time çalışan sayısı çok azdır.

Farklılıklar:

- **$\leq 50K$ gelir grubu**, çalışma saatlerinde daha geniş bir dağılıma sahiptir.
- **$>50K$ gelir grubu** ise daha sıkı bir dağılım gösterir ve genellikle tam zamanlı çalışan bireylerden oluşur.

Benzerlikler:

- Her iki grupta da **40 saat civarı** çalışma süreleri yoğunlaşmıştır.

Grafik, gelir gruplarına göre çalışma saatleri dağılımındaki farklılıkları net bir şekilde ortaya koymaktadır:

- **$\leq 50K$ Gelir Grubu:** Part-time çalışanların daha yaygın olduğu görülmektedir.
- **$>50K$ Gelir Grubu:** Tam zamanlı çalışan bireylerin baskın olduğu ve gelir artışıyla çalışma sürelerinin odaklandığı anlaşılmaktadır.

Çıkarımlar:

- Çalışma saatlerinin düzenli olması (>40 saat) daha yüksek gelir gruplarında daha yaygındır.
- Part-time çalışanlar genellikle düşük gelir grubunda yer almaktadır.
- Çalışma saatleri, gelir tahmini için önemli bir özellik olarak değerlendirilebilir.

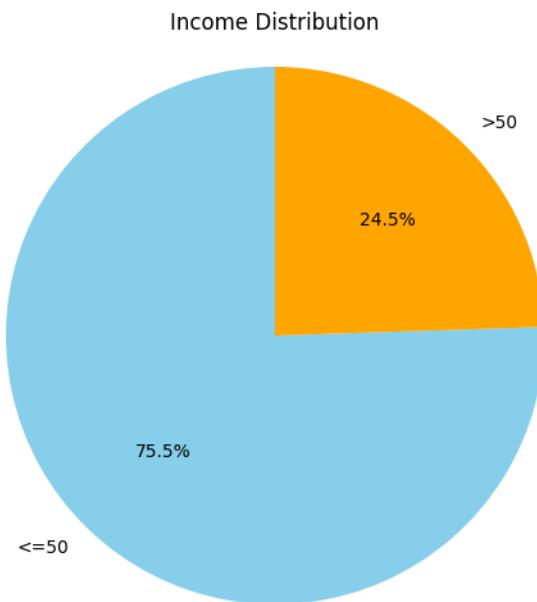
4.10 Gelir Dağılımı Analizi (Pasta Grafiği)

Bu kodlar, 'income' sütunundaki gelir seviyelerini yüzdelik olarak hesaplayıp, her seviyeyi pasta grafiğiyle görselleştirecek, bireylerin gelir gruplarına dağılımını analiz etmek için kullanılmıştır.

```
# Gelir dağılımı yüzdesini hesaplama
income_distribution = df_del['income'].value_counts(normalize=True)

# Gelir kategorileri
income_labels = ['<=50', '>50']

# Pasta grafiği çizimi
plt.figure(figsize=(8, 6))
plt.pie(income_distribution, labels=income_labels, autopct='%.1f%%', startangle=90, colors=['skyblue', 'orange'])
plt.title('Income Distribution')
plt.axis('equal') # Pasta grafiğini daire olarak tutar
plt.show()
```



ELDE EDİLEN SONUÇLAR:

≤ 50 Gelir Grubu:

- Veri setindeki bireylerin **75.5%**'i $\leq 50K$ gelir seviyesinde yer almaktadır.
- Bu grup, veri setinde baskındır ve düşük gelir grubunu temsil eder.

>50 Gelir Grubu:

- Veri setindeki bireylerin **24.5%**'i $>50K$ gelir seviyesindedir.
- Bu grup, daha az temsil edilen yüksek gelir grubudur.

Gruplar Arası Dengesizlik:

- Gelir seviyeleri arasında belirgin bir dengesizlik var:
 - ≤ 50 Gelir Grubu**, toplamın yaklaşık dörtte üçünü oluşturuyor.
 - >50 Gelir Grubu** ise yalnızca dörtte bir oranında temsil ediliyor.

Grafik, gelir seviyelerinin dağılımını net bir şekilde göstermektedir:

- **≤50 Gelir Grubu**, toplam bireylerin yaklaşık üçte ikisini oluşturarak baskın gruptur.
- **>50 Gelir Grubu** ise daha küçük bir dilime sahiptir ve veri setindeki azınlık grubunu temsil eder.
- **Dengesizlik**, veri setinde yüksek gelir grubunun azınlıkta olduğunu ortaya koymaktadır.

Çıkarımlar:

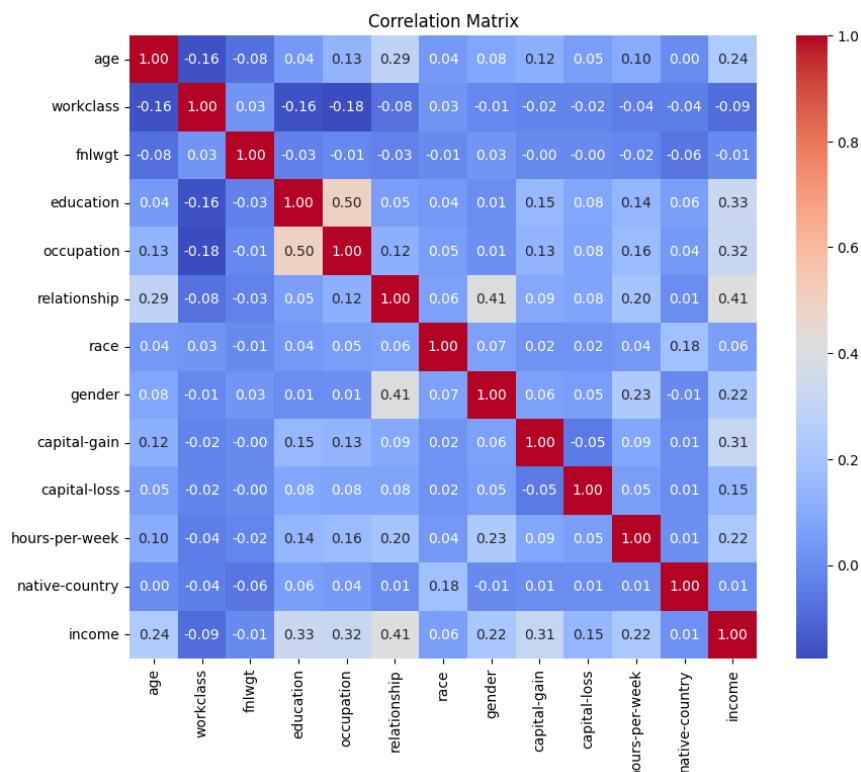
- Veri seti dengesizdir, çünkü düşük gelir grubu (≤ 50) açık ara baskın durumdadır.
- Bu durum, veri modellemesi yapılrken dikkat edilmesi gereken bir unsurdur.
 - Örneğin, **sınıf dengesizliği**, tahmin modellerinde düşük gelir grubuna fazla ağırlık verilmesine neden olabilir.

4.11 Korelasyon Matrisi Analizi

Korelasyon matrisi, veri setindeki sayısal sütunlar arasındaki ilişkileri analiz etmek için oluşturulmuştur. Bu matrisi kullanarak hangi değişkenlerin birbirileyle güçlü, zayıf, pozitif veya negatif ilişki içinde olduğunu görebiliriz. Özellikle **income (gelir)** değişkeni ile diğer değişkenler arasındaki korelasyonları inceleyerek, gelir tahmini yaparken hangi değişkenlerin daha etkili olduğunu belirlemek ve modelleme sürecinde hangi özelliklere öncelik verilmesi gerektiğini anlamak amaçlanmıştır.

```
# Compute the correlation matrix for numerical columns
correlation_matrix = df_del.corr()

# Plot the correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



ELDE EDİLEN SONUÇLAR:

Gelir ile En Yüksek Korelasyon Gösteren Değişkenler:

- **Eğitim (education):** Gelirle en yüksek pozitif korelasyona sahip sütunlardan biridir (**0.33**). Eğitim seviyesi arttıkça gelirin artma olasılığı yüksektir.
- **İlişki Durumu (relationship):** Gelirle anlamlı bir pozitif korelasyona sahiptir (**0.41**). Ailevi veya bireysel durumlar gelir üzerinde etkili olabilir.
- **Meslek (occupation):** Gelirle pozitif bir korelasyona sahiptir (**0.32**). Bazı meslek grupları daha yüksek gelir sağlar.

Negatif veya Düşük Korelasyon Gösteren Değişkenler:

- **Çalışma Saatleri (hours-per-week):** Gelirle düşük pozitif bir korelasyona sahiptir (**0.22**). Çalışma saatleri geliri etkiler ancak bu etki sınırlıdır.
- **Yaş (age):** Gelirle pozitif bir ilişki gösterse de (**0.24**) bu ilişki güçlü değildir. Belirli bir yaşın üzerinde gelir artışı durabilir.
- **Workclass:** Gelirle düşük negatif bir korelasyona sahiptir (-**0.09**). Bu, çalışma sınıfları arasındaki gelir farklılıklarını yansıtabilir.

Sütunlar Arasındaki Diğer İlişkiler:

- **Eğitim ve Meslek (education & occupation):** Bu iki değişken arasında orta düzeyde pozitif bir ilişki vardır (**0.50**). Eğitim seviyesi arttıkça, daha yüksek gelir sağlayan mesleklerin seçilmesi olasıdır.
- **Cinsiyet ve İlişki Durumu (gender & relationship):** Aralarında anlamlı bir pozitif ilişki vardır (**0.41**). Cinsiyetin ailevi durumu belirlemede etkili olduğu düşünülebilir.

Korelasyon matrisindeki kırmızı tonlar, değişkenler arasında pozitif bir ilişki olduğunu; mavi tonlar ise negatif bir ilişki olduğunu gösterir.

Gelir (income) ile diğer sütunlar arasındaki ilişki genel olarak pozitif olsa da, hiçbir korelasyon katsayıısı 0.50'nin üzerinde değildir. Bu, gelir üzerindeki etkilerin çok faktörlü olduğunu ve diğer değişkenlerle ilişkilerinin sınırlı olabileceğini göstermektedir.

4.12 Değişim Katsayısı (Coefficient of Variation) Analizi

Varyasyon katsayısını hesaplamamızın amacı, her değişkenin kendi ortalamasına göre ne kadar değişkenlik gösterdiğini anlamaktır. Bu analiz, özellikle **hedef değişken olan gelir (income)** ile ilişkilendirilecek değişkenleri seçerken önemlidir. Yüksek varyasyon katsayısına sahip değişkenler, genellikle daha fazla bilgi içerebilir ve modele katkı sağlayabilir. Öte yandan, düşük varyasyon katsayısına sahip değişkenler daha sabit bir dağılıma sahip olduğu için gelir tahmini açısından daha az ayırt edici olabilir. Bu nedenle, hangi değişkenlerin gelir tahmini için daha açıklayıcı olabileceğini belirlemek adına varyasyon katsayısı kritik bir metrik olarak kullanılmıştır.

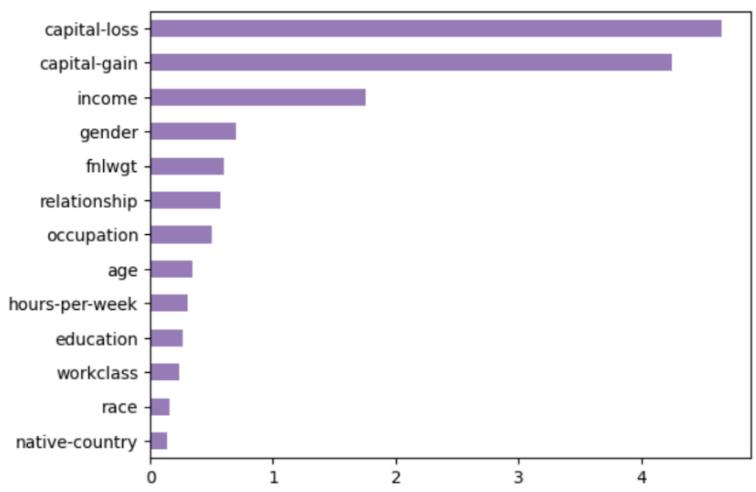
```

coefficient_of_variation = df_del.std() / df_del.mean()
print(coefficient_of_variation*100)

sorted_cv = coefficient_of_variation.sort_values()
sorted_cv.plot.barh(x = 'Variables', y = 'Coefficient of Variation', color = '#967bb6')

```

age	34.307003
workclass	23.130042
fnlwgt	59.946104
education	26.938794
occupation	50.615930
relationship	56.739928
race	16.075662
gender	69.498313
capital-gain	424.804197
capital-loss	465.899127
hours-per-week	30.024235
native-country	14.000457
income	175.608611
dtype: float64	
<Axes: >	



ELDE EDİLEN SONUÇLAR:

En Yüksek Değişim Gösteren Değişkenler:

- **Capital-loss ve capital-gain:** Bu değişkenler en yüksek değişim katsayısına sahiptir (yaklaşık %424 ve %465). Bu, bu değişkenlerin ortalamalarına kıyasla oldukça büyük varyasyona sahip olduğunu gösterir.
 - **Neden?** Bu değişkenlerin çoğu birey için sıfır olabileceği, ancak bazı bireylerde yüksek değerlere sahip olduğu için büyük varyasyon göstermektedir.

Düşük Değişim Gösteren Değişkenler:

- **Native-country, race, ve workclass:** Bu değişkenler en düşük değişim katsayısına sahiptir (yaklaşık %14-%23). Bu, bu değişkenlerin daha sabit ve ortalama değerlerine daha yakın olduğunu gösterir.
 - **Neden?** Bu değişkenler genellikle kategorik özelliklere sahiptir ve dağılımları daha homojendir.

Orta Düzey Değişim Gösteren Değişkenler:

- **Income, gender, ve relationship** gibi değişkenler orta düzeyde değişim göstermektedir.

- Gelir değişkeninin %175 civarında bir değişim katsayısı vardır, bu da gelir grupları arasında belirgin bir varyasyon olduğunu ifade eder.

Grafik, değişim katsayılarını net bir şekilde sıralamaktadır.

Capital-loss ve **capital-gain**, yüksek varyasyonlarıyla diğer değişkenlerden ayırmaktadır.

Native-country gibi değişkenler ise düşük varyasyonlarıyla daha stabil olarak görülmektedir.

4.13 Çarpıklık (Skewness) ve Basıklık (Kurtosis) Analizi

Bu kodlar, veri setindeki değişkenlerin **çarpıklık (skewness)** ve **basıklık (kurtosis)** analizini yaparak, değişkenlerin dağılım özelliklerini değerlendirmek, normal dağılıma uygunluklarını test etmek ve anormal durumları tespit etmek için kullanılmıştır. Bu analiz, modelleme öncesinde veri setinin yapısını anlamak ve gerekli ön işlemleri belirlemek adına kritik bir adımdır.

```
from tabulate import tabulate
from scipy.stats import skew, kurtosis, normaltest, shapiro

skewness = df_del.skew()
kurtosisness = df_del.kurtosis()

headers = ['Variable', 'Shapiro Test', 'Skewness C.', 'Skewness', 'Kurtosis C.', 'Kurtosis']

table_data = []
for col in df_del.columns:
    stat, p = normaltest(df_del[col])
    skewness_val = f'{skewness[col]:.4f}'
    skewness_desc = 'High skewness' if skewness[col] < -1 or skewness[col] > 1 else 'Symmetric or slightly skewed'
    kurtosis_val = f'{kurtosisness[col]:.4f}'
    kurtosis_desc = 'Very flat or peaked' if kurtosisness[col] < -2 or kurtosisness[col] > 2 else 'Normal or slightly flat'

    stat_sh, p_sh = shapiro(df_del[col])
    shapiro_test = 'Yes' if p_sh >= 0.05 else 'No'

    table_data.append([col, shapiro_test, skewness_val, skewness_desc, kurtosis_val, kurtosis_desc])

table = tabulate(table_data, headers, tablefmt='pretty')
print(table)
```

Variable	Shapiro Test	Skewness C.	Skewness	Kurtosis C.	Kurtosis
age	No	0.5378	Symmetric or slightly skewed	-0.1498	Normal or slightly flat
workclass	No	-1.6973	High skewness	1.5147	Normal or slightly flat
fnlwgt	No	1.4537	High skewness	6.2722	Very flat or peaked
education	No	-0.1392	Symmetric or slightly skewed	0.3551	Normal or slightly flat
occupation	No	-0.1789	Symmetric or slightly skewed	-1.0108	Normal or slightly flat
relationship	No	-0.1403	Symmetric or slightly skewed	-1.6291	Normal or slightly flat
race	No	-3.6299	High skewness	14.9018	Very flat or peaked
gender	No	-0.7439	Symmetric or slightly skewed	-1.4466	Normal or slightly flat
capital-gain	No	5.5786	High skewness	36.9807	Very flat or peaked
capital-loss	No	4.5228	High skewness	18.8431	Very flat or peaked
hours-per-week	No	0.3354	Symmetric or slightly skewed	3.2332	Very flat or peaked
native-country	No	-4.5183	High skewness	22.2939	Very flat or peaked
income	No	1.1867	High skewness	-0.5919	Normal or slightly flat

ELDE EDİLEN SONUÇLAR:

Shapiro Test:

Shapiro-Wilk testi, değişkenlerin normal dağılıma uygunluğunu test eder:

- Yes:** Değişken normal dağılıma uygundur ($p \geq 0.05$).
- No:** Değişken normal dağılıma uygun değildir ($p < 0.05$).

Çıkarım: Çıktıyla göre hiçbir değişken normal dağılıma uymamaktadır (tüm değerler "No").

Skewness C. (Çarpıklık Katsayısı):

Bu sütun, çarpıklık katsayısını gösterir:

- Pozitif çarpıklık: Dağılım sağa doğru uzun bir kuyruğa sahiptir.
- Negatif çarpıklık: Dağılım sola doğru uzun bir kuyruğa sahiptir.
- 0'a yakın değerler: Simetrik veya çarpıklığı az olan dağılımlardır.

Örnekler:

- **capital-gain (5.5786)** ve **capital-loss (4.2228)** gibi değişkenlerde yüksek pozitif çarpıklık vardır. Bu, bu değişkenlerin sağ kuyruğunda aşırı değerler olduğunu gösterir.
- **workclass (-1.6973)** değişkeni yüksek negatif çarpıklık sergiler, yani sola doğru uzamış bir dağılıma sahiptir.

Skewness (Çarpıklık Durumu):

Çarpıklık katsayısının sözel yorumu:

- **High skewness:** Çarpıklık katsayısı $|1|$ 'den büyük, yani asimetrik bir dağılım vardır.
- **Symmetric or slightly skewed:** Çarpıklık katsayısı $|1|$ 'den küçük, dağılım simetrik veya hafif çarpıktır.

Örnekler:

- **gender** ve **age:** Simetrik veya hafif çarpık (daha dengeli).
- **capital-gain** ve **capital-loss:** Yüksek çarpıklığa sahiptir, veri dağılımında ciddi asimetri vardır.

Kurtosis C. (Basıklık Katsayısı):

Bu sütun, basıklık katsayısını gösterir:

- Pozitif basıklık: Dağılım sivri tepelidir (daha keskin bir zirve).
- Negatif basıklık: Dağılım daha basık ve yayılmıştır.
- 0'a yakın değerler: Normal dağılıma yakındır.

Örnekler:

- **capital-gain (36.9807)** ve **capital-loss (18.8431)** çok yüksek basıklık değerlerine sahiptir. Bu, dağılımin tepe noktasının çok sivri olduğunu ve uç değerlerin etkili olduğunu gösterir.
- **age (-0.1498)** gibi değişkenler ise basıklık açısından normale daha yakındır.

Kurtosis (Basıklık Durumu):

Basıklık katsayısının sözel yorumu:

- **Very flat or peaked:** Katsayı $|2|$ 'den büyük, dağılım çok sivri veya çok basıktır.
- **Normal or slightly flat:** Katsayı $|2|$ 'ye yakın, dağılım normal veya hafif basıktır.

Örnekler:

- **capital-gain** ve **capital-loss**: "Very flat or peaked" (sivri veya uç değerler çok baskın).
- **age**, **relationship**, **gender** gibi değişkenler: "Normal or slightly flat" (daha dengeli bir dağılım).

GENEL YORUM:

Normal Dağılım:

- Shapiro-Wilk testine göre hiçbir değişken normal dağılıma uymuyor. Bu, parametrik analizler veya modelleme yapılmadan önce veri dönüşümlerine ihtiyaç duyulabileceğini gösteriyor.

Çarpıklık:

- **Capital-gain**, **capital-loss**, ve **race** gibi değişkenler yüksek çarpıklığa sahiptir. Bu durum, bu değişkenlerin uç değerler barındırdığını ve dağılımda asimetri olduğunu gösterir.

Basıklık:

- Yüksek basıklık değerleri (örneğin **capital-gain**), uç değerlerin etkisinin güçlü olduğunu ifade eder. Bu durum, modelleme sürecinde dikkate alınmalıdır.

Etkisi Olabilecek Değişkenler:

- **capital-gain**, **capital-loss**, ve **native-country** gibi değişkenler, yüksek çarpıklık ve basıklık değerleriyle öne çıkar.

5.Veri Modelleme

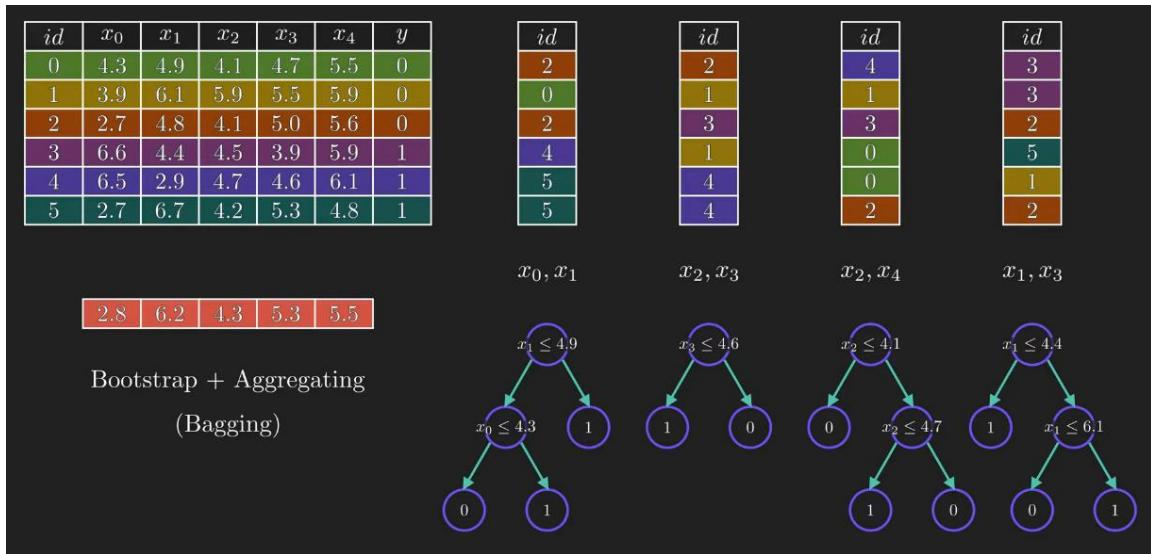
5.1 Algoritma Seçimi

Bireylerin belirli demografik ve sosyo-ekonomik özelliklerine dayanarak yıllık gelirlerinin 50.000 birimden yüksek veya düşük olduğunu tahmin etmek hedefleniyor. Yani problem ikili sınıflandırma problemi. Bu sebeple probleme uygun algoritmalar araştırıldı.

1. Random Forest Classifier

Eğitim verilerine karşı çok hassas olduğundan yüksek varyansa yol açan karar ağaçlarının birden fazla ve rastgele şekilde bir araya gelmesiyle oluşan bir koleksiyondur. Her bir karar ağaçının tahminine dayanarak bir sınıf veya değer çıkarmak için çoğuluk oyu (sınıflandırma) alır. Nihai tahmin için ağaçlardan tahmin değerleri talep edilirken her bir ağaçın daha önce hesaplanan hata oranları göz önüne alınarak ağaçlara ağırlık verilir. Çok başarısız olan ağaçların tahmin değerlerini neredeyse önemsemeyez.

- Bootstrap (Rastgele alt örneklem)
- Özellik alt kümesi seçimi
- Karar ağaçları oluşturma
- Tahmin (Ensemble yaklaşımı) şeklinde çalışır.



2. Logistic Regression:

Logistic Regression, doğrusal regresyonun temelini alır ancak sonuçları doğrusal değil, sınıflar arasında bir olasılık tahmini yapmak için logistic (sigmoid) fonksiyonu kullanır. Bir sınıflandırma algoritmasıdır ve bağımlı değişkenin kategorik olduğu (genellikle ikili 0 ya da 1) problemler için kullanılır. Logistic Regression, sınıflar arasında doğrusal bir sınır olduğunu varsayar. Eğer sınıflar doğrusal olmayan bir şekilde ayrılıyorsa (ki mevcut problem için doğrusal değil) modelin performansı zayıf olabilir. Bu sebeple logistic regression tercih edilmedi

3. XGBoost:

XGBoost, karar ağacı temelli ve eğim-artırmalı (gradient boost) bir makine öğrenmesi sistemidir. Mevcut sınıflandırma problemine uygun denetimli öğrenme (supervised learning) algoritmasıdır. Karmaşık karar sınırlarını öğrenme yeteneğine sahip olduğundan, doğrusal olmayan ve karmaşık veri setlerinde iyi performans gösterir. Model, hangi özelliklerin tahminleme sürecine daha fazla katkı sağladığını belirleyebilir. Veri dağılımı dengesiz ise çok daha etkili olabilir. Bunun nedeni, class weights veya sampling tekniklerini desteklemesi ve modelin her iki sınıfı da etkili bir şekilde öğrenmeye çalışmasıdır.

Bu özelliklerin veri setine uygun olduğuna karar verildi ve XGBoost tercih edildi.

5.2 Model Eğitimi

Öncelikle veri seti eğitim ve test olmak üzere ikiye ayrıldı.

```
[197] from sklearn.model_selection import train_test_split

# Hedef değişken ve özellikler
X = df_deli.drop(columns=['income'])
y = df_deli['income']

# Veriyi %80 eğitim ve %20 test olarak ayırmaya işlemi
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Sonuçları kontrol etme
print("Eğitim veri boyutu: {} \nTest veri boyutu: {}".format(X_train.shape, X_test.shape))
print("Eğitim setindeki income dağılımı: \n{}\nTest setindeki income dağılımı: \n{}".format(y_train.value_counts(normalize=True), y_test.value_counts(normalize=True)))

#> Eğitim veri boyutu: (35607, 12)
#> Test veri boyutu: (8902, 12)
#> Eğitim setindeki income dağılımı:
#> income
#> 0    0.755132
#> 1    0.244868
#> Name: proportion, dtype: float64
#> Test setindeki income dağılımı:
#> income
#> 0    0.755111
#> 1    0.244889
#> Name: proportion, dtype: float64
```

Veri setinin doğruluğunu değerlendirebilmek için K-Fold Cross Validation yöntemi kullanıldı. Bu yöntem, modelin genellenebilirliğini değerlendirme açısından oldukça önemlidir, çünkü tüm veri seti hem eğitim hem de test seti olarak birden fazla kez kullanılır. Bu sayede modelin overfitting (aşırı uyum) yapması engellenebilir ve modelin performansı daha güvenilir bir şekilde ölçülebilir. Bu kodda katlama işleminin sınıf dağılımları ve veri bölünmesi analiz edilir.

```
[ ] from sklearn.model_selection import StratifiedKFold

# Stratified K-Fold oluşturma
kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# K-Fold içinde eğitim ve validasyon gruplarını inceleme
for fold, (train_idx, val_idx) in enumerate(kf.split(X_train, y_train)):
    print(f"Fold {fold+1}")
    print(f"Eğitim seti boyutu: {len(train_idx)}")
    print(f"Validasyon seti boyutu: {len(val_idx)}")
    print(f"Eğitim setindeki income dağılımı:\n{y_train.iloc[train_idx].value_counts(normalize=True)}")
    print(f"Validasyon setindeki income dağılımı:\n{y_train.iloc[val_idx].value_counts(normalize=True)}\n")
    print("-" * 30)

    ➔ Fold 1
    Eğitim seti boyutu: 32046
    Validasyon seti boyutu: 3561
    Eğitim setindeki income dağılımı:
    income
    0    0.755133
    1    0.244867
    Name: proportion, dtype: float64
    Validasyon setindeki income dağılımı:
    income
    0    0.755125
    1    0.244875
    Name: proportion, dtype: float64

    -----
    Fold 2
    Eğitim seti boyutu: 32046
    Validasyon seti boyutu: 3561
    Eğitim setindeki income dağılımı:
    income
    0    0.755133
    1    0.244867
    Name: proportion, dtype: float64
    Validasyon setindeki income dağılımı:
    income
    0    0.755125
    1    0.244875
    Name: proportion, dtype: float64
```

Analiz sonrasında model eğitildi.

```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Model oluşturma
model = XGBClassifier(random_state=42)
fold_metrics = []

for fold, (train_idx, val_idx) in enumerate(kf.split(X_train, y_train)):
    print(f"Fold {fold+1}")

    # Eğitim ve validasyon verilerini ayır
    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    # Modeli eğit
    model.fit(X_train_fold, y_train_fold)
```

6. Model Değerlendirme

Model her fold için ayrı ayrı eğitilir ve test edilir, böylece modelin genel genelleme performansı daha doğru bir şekilde analiz edilir. Bu çalışmada, veri setimiz 10 katlama (10-fold) ayrılmıştır. Her bir katlama için:

1. **Model Eğitimi:** Eğitim verileri üzerinde model eğitilmiştir.
2. **Tahmin:** Validasyon verileri üzerinde model tahmin yapmıştır.
3. **Performans Analizi:** Aşağıdaki metrikler hesaplanmıştır:
 - a. **Doğruluk (Accuracy):** Tüm sınıflar için doğru tahminlerin oranıdır.
 - b. **Kesinlik (Precision):** Pozitif sınıfın doğruluğunu ifade eder.
 - c. **Duyarlılık (Recall):** Pozitif sınıfın ne kadar iyi tahmin edildiğini gösterir.
 - d. **F1 Skoru (F1 Score):** Kesinlik ve duyarlılığın harmonik ortalamasıdır.

```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Model oluşturma
model = XGBClassifier(random_state=42)
fold_metrics = []

for fold, (train_idx, val_idx) in enumerate(kf.split(X_train, y_train)):
    print(f"Fold {fold+1}")

    # Eğitim ve validasyon verilerini ayı
    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    # Modeli eğit
    model.fit(X_train_fold, y_train_fold)

    # Validasyon seti Üzerinde tahmin yap
    y_val_pred = model.predict(X_val_fold)

    # Performans metriklerini hesapla
    accuracy = accuracy_score(y_val_fold, y_val_pred)
    precision = precision_score(y_val_fold, y_val_pred, average='binary')
    recall = recall_score(y_val_fold, y_val_pred, average='binary')
    f1 = f1_score(y_val_fold, y_val_pred, average='binary')
    fold_metrics.append({'accuracy': accuracy, 'precision': precision, 'recall': recall, 'f1': f1})

    # Sonuçları yazdır
    print(f" Fold {fold+1} doğruluk skoru (Accuracy): {accuracy:.4f}")
    print(f" Fold {fold+1} kesinlik (Precision): {precision:.4f}")
    print(f" Fold {fold+1} duyarlılık (Recall): {recall:.4f}")
    print(f" Fold {fold+1} F1 Skoru: {f1:.4f}")
    print("-" * 30)

# Tüm fold'ların sonuçlarının analizi
print("Cross Validation Sonuçları:")
average_metrics = {
    'accuracy': sum(metric['accuracy'] for metric in fold_metrics) / len(fold_metrics),
    'precision': sum(metric['precision'] for metric in fold_metrics) / len(fold_metrics),
    'recall': sum(metric['recall'] for metric in fold_metrics) / len(fold_metrics),
    'f1': sum(metric['f1'] for metric in fold_metrics) / len(fold_metrics),
}

print(f"Ortalama doğruluk skoru (Accuracy): {average_metrics['accuracy']:.4f}")
print(f"Ortalama kesinlik (Precision): {average_metrics['precision']:.4f}")
print(f"Ortalama duyarlılık (Recall): {average_metrics['recall']:.4f}")
print(f"Ortalama F1 Skoru: {average_metrics['f1']:.4f})
```

```

Fold 1
Fold 1 doğruluk skoru (Accuracy): 0.8742
Fold 1 kesinlik (Precision): 0.7782
Fold 1 duyarlılık (Recall): 0.6800
Fold 1 F1 Skoru: 0.7258

Fold 2
Fold 2 doğruluk skoru (Accuracy): 0.8675
Fold 2 kesinlik (Precision): 0.7817
Fold 2 duyarlılık (Recall): 0.6365
Fold 2 F1 Skoru: 0.7016

Fold 3
Fold 3 doğruluk skoru (Accuracy): 0.8697
Fold 3 kesinlik (Precision): 0.7757
Fold 3 duyarlılık (Recall): 0.6583
Fold 3 F1 Skoru: 0.7122

Fold 4
Fold 4 doğruluk skoru (Accuracy): 0.8644
Fold 4 kesinlik (Precision): 0.7625
Fold 4 duyarlılık (Recall): 0.6479
Fold 4 F1 Skoru: 0.7006

Fold 5
Fold 5 doğruluk skoru (Accuracy): 0.8618
Fold 5 kesinlik (Precision): 0.7676
Fold 5 duyarlılık (Recall): 0.6250
Fold 5 F1 Skoru: 0.6890

Fold 6
Fold 6 doğruluk skoru (Accuracy): 0.8627
Fold 6 kesinlik (Precision): 0.7543
Fold 6 duyarlılık (Recall): 0.6514
Fold 6 F1 Skoru: 0.6991

Fold 6
Fold 6 doğruluk skoru (Accuracy): 0.8627
Fold 6 kesinlik (Precision): 0.7543
Fold 6 duyarlılık (Recall): 0.6514
Fold 6 F1 Skoru: 0.6991

Fold 7
Fold 7 doğruluk skoru (Accuracy): 0.8655
Fold 7 kesinlik (Precision): 0.7652
Fold 7 duyarlılık (Recall): 0.6502
Fold 7 F1 Skoru: 0.7030

Fold 8
Fold 8 doğruluk skoru (Accuracy): 0.8640
Fold 8 kesinlik (Precision): 0.7523
Fold 8 duyarlılık (Recall): 0.6625
Fold 8 F1 Skoru: 0.7045

Fold 9
Fold 9 doğruluk skoru (Accuracy): 0.8683
Fold 9 kesinlik (Precision): 0.7683
Fold 9 duyarlılık (Recall): 0.6617
Fold 9 F1 Skoru: 0.7110

Fold 10
Fold 10 doğruluk skoru (Accuracy): 0.8677
Fold 10 kesinlik (Precision): 0.7773
Fold 10 duyarlılık (Recall): 0.6445
Fold 10 F1 Skoru: 0.7047

Cross Validation Sonuçları:
Ortalama doğruluk skoru (Accuracy): 0.8666
Ortalama kesinlik (Precision): 0.7683
Ortalama duyarlılık (Recall): 0.6518
Ortalama F1 Skoru: 0.7052

```

6.1 Overfitting ve Underfitting Kontrolü

Modelin eğitim verilerindeki detaylara ve gürültülere fazla uyum sağlama durumunda ortaya çıkar. Bu durumda model:

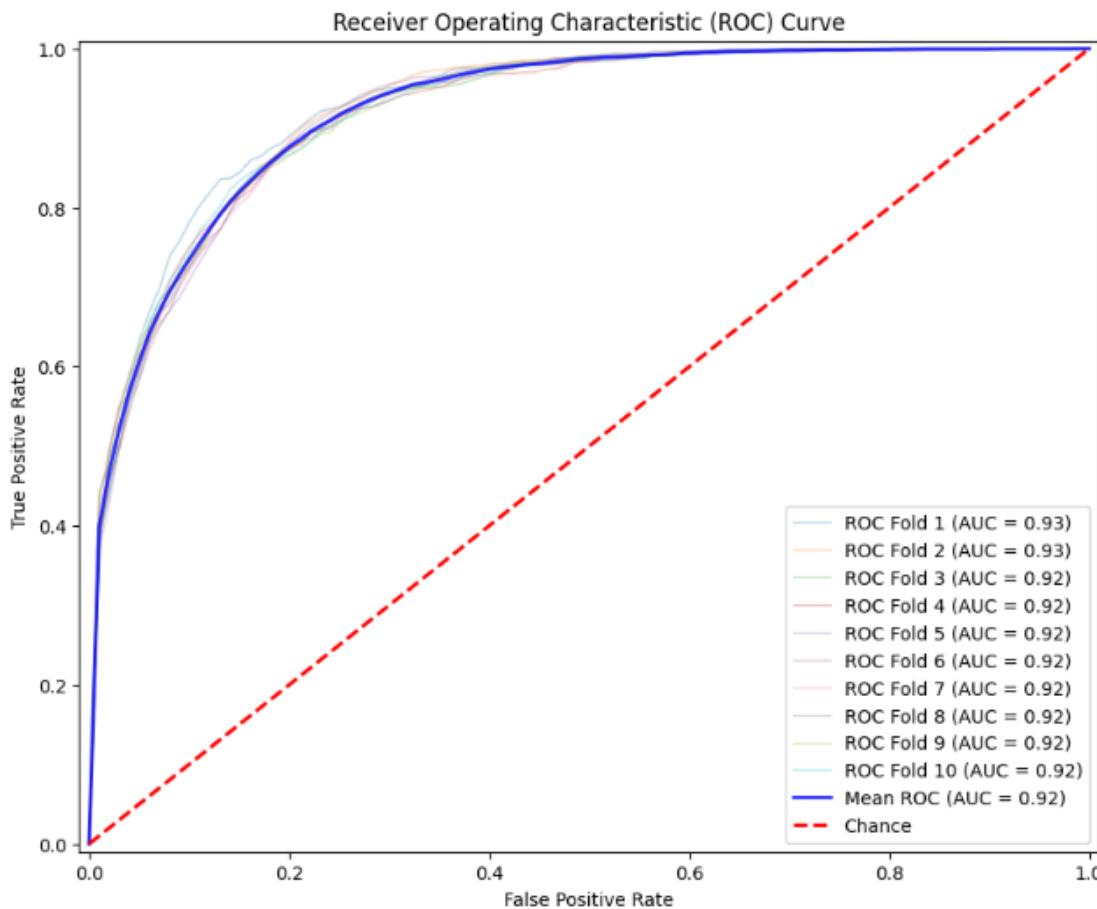
- Eğitim setinde yüksek performans gösterirken,
- Validasyon veya test setinde düşük performans sergiler.

Bu durum, modelin yeni verilere genelleme yapma yeteneğinin zayıf olduğunu gösterir.

K-Fold Cross Validation yöntemi, overfitting'i kontrol etmek etkili bir yöntemdir çünkü:

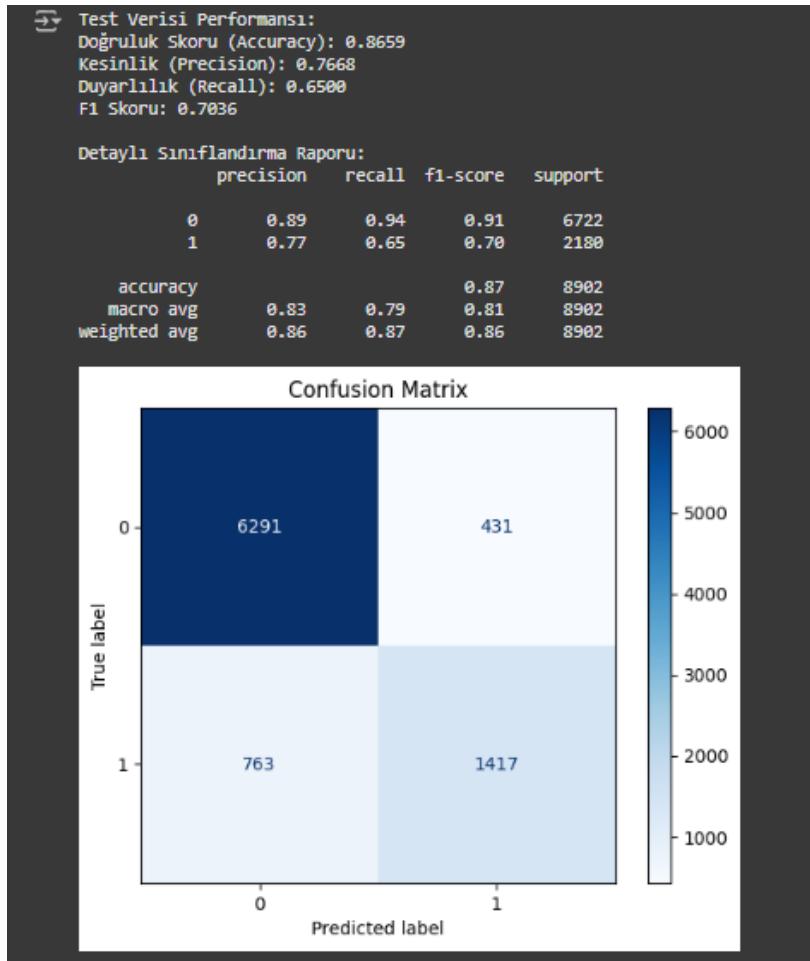
- Eğitim ve validasyon setleri sürekli olarak değişir.
- Model her bir fold'da farklı veri setlerinde test edilir.
- Eğer model, tüm fold'larda tutarlı bir şekilde yüksek performans sergiliyorsa, bu genelleme yeteneğinin iyi olduğunu gösterir ve overfitting riskini azaltır.

7. Sonuçların Yorumlanması ve Görselleştirilmesi



- ROC Eğrisi Açıklaması:** ROC eğrisi, sınıflandırma modelinin performansını ölçmek için kullanılır. Eğri, True Positive Rate (TPR) (Duyarlılık) ve False Positive Rate (FPR) değerlerini farklı eşik değerleri için grafiğe döker. Model, ideal olarak, sol üst köşeye yakın bir eğri çizmeli ve yüksek bir AUC değerine sahip olmalıdır.
- Elde Edilen Sonuçlar:** Grafikte, her bir fold için ROC eğrileri gösterilmektedir. Her fold'un AUC değeri yaklaşık 0.92-0.93 arasında değişmektedir. Bu, modelin tutarlı bir şekilde iyi performans gösterdiğini, sınıflandırma başarısının yüksek olduğunu göstermektedir.
- Ortalama ROC Eğrisi:** Mavi kalın çizgi, tüm fold'ların ortalaması ROC eğrisini göstermektedir. Ortalama AUC değeri 0.92 olarak hesaplanmıştır. Bu, modelin genel olarak yüksek bir ayrıştırma gücüne sahip olduğunu ve pozitif ve negatif sınıflar arasında iyi bir ayırım yaptığıni ifade eder.
- Kırmızı Noktalı Çizgi:** Bu çizgi, rastgele tahminlerin performansını ($AUC = 0.50$) temsil eder. Modelin tüm fold'lar için AUC değerinin bu çizginin oldukça üzerinde olması, modelin rastgele tahminlerden çok daha iyi olduğunu kanıtlamaktadır.

Bu ROC eğrileri, modelin overfitting veya underfitting yaşamadığını göstermektedir. Model, her fold üzerinde tutarlı bir şekilde iyi bir performans sergilemiştir ve farklı veri dağılımlarında genel olarak başarılı olma eğilimindedir. Bu durum, modelin hem eğitim hem de test verileri üzerinde iyi bir genelleme yapabildiğini ifade eder.



- Doğruluk Skoru (Accuracy):**
- Modelin doğruluk skoru 0.8659, yani test verilerindeki örneklerin %86.59'u doğru sınıflandırılmıştır. Bu, modelin genel olarak iyi bir performans sergilediğini gösterir.
- Kesinlik (Precision):**
- Sınıf 0 için:** 0.89 (modelin negatif tahminlerinde %89 doğruluk).
- Sınıf 1 için:** 0.77 (modelin pozitif tahminlerinde %77 doğruluk).
- Pozitif sınıf için kesinlik nispeten düşük, bu da modelin yanlış pozitiflere (False Positives) dikkat etmesi gerektiğini gösterir.
- Duyarlılık (Recall):**
- Sınıf 0 için:** 0.94 (negatif sınıfların %94'ü doğru tahmin edilmiş).
- Sınıf 1 için:** 0.65 (pozitif sınıfların sadece %65'i doğru tahmin edilmiş).
- Pozitif sınıfta duyarlılığın daha düşük olması, modelin bazı pozitif örnekleri atladığını (False Negatives) gösterir.
- F1 Skoru:**
- F1 skoru (kesinlik ve duyarlılığın harmonik ortalaması) genel olarak dengeli bir değerlendirme sağlar.
- Sınıf 0 için:** 0.91
- Sınıf 1 için:** 0.70
- Pozitif sınıfda F1 skoru, modelin bu sınıfda iyileştirmeye ihtiyaç duyduğunu vurgulamaktadır.
- Destek (Support):**

- Sınıf 0: 6722 örnek
 - Sınıf 1: 2180 örnek
 - Veri setinde sınıflar arasında bir dengesizlik olduğu görülmektedir. Bu durum model performansını etkileyebilir.
-
- **True Negatives (TN):** 6291 (Model, 0 sınıfını doğru tahmin etmiş).
 - **False Positives (FP):** 431 (Model, 1 olarak tahmin ettiği ancak gerçekte 0 olan örnekler).
 - **False Negatives (FN):** 763 (Model, 0 olarak tahmin ettiği ancak gerçekte 1 olan örnekler).
 - **True Positives (TP):** 1417 (Model, 1 sınıfını doğru tahmin etmiş).

Confusion matrix, modelin negatif sınıfları daha iyi tahmin ettiğini (TN ve TP yüksek), ancak pozitif sınıfta daha fazla hata yaptığını (FP ve FN) ortaya koymaktadır.

KAYNAKÇA

https://digitalcommons.bryant.edu/cgi/viewcontent.cgi?article=1038&context=honors_economics

https://escholarship.org/content/qt6d01c9v7/qt6d01c9v7_noSplash_d6307f10bef85009fad51d5837b90bb1.pdf?t=seap4w

<https://www.geeksforgeeks.org/stratified-k-fold-cross-validation/>

[https://www.kaggle.com/code/beyzacankurtaran/aygaz-data-analysis-bootcamp#7\)-Model-Creation-and-Model-Results](https://www.kaggle.com/code/beyzacankurtaran/aygaz-data-analysis-bootcamp#7)-Model-Creation-and-Model-Results)

<https://www.kaggle.com/datasets/wenruliu/adult-income-dataset/data>

<https://medium.com/@melisacevik13/ke%C5%9Fif%C3%A7i-veri-analizi-edanedir-1c0fecede74>

<https://www.kaggle.com/code/hafshawahab/employee-income-analysis>

https://tr.wikipedia.org/wiki/Varyasyon_katsay%C4%B1s%C4%B1

https://www.youtube.com/watch?v=v6VJ2RO66Ag&ab_channel=NormalizedNerd

<https://medium.com/yaz%C4%B1l%C4%B1m-ve-bili%C5%9Fim-kul%C3%BCb%C3%BC/random-forests-92fd17d9aa4f>

<https://link.springer.com/article/10.1023/a:1010933404324>

https://www.youtube.com/watch?v=FN5z4iPFHzw&ab_channel=BoraCanbula

<https://www.sciencedirect.com/science/article/pii/S1532046403000340>

https://www.youtube.com/watch?v=YMJtsYIp4kg&ab_channel=CodeEmporium

https://link.springer.com/chapter/10.1007/978-3-030-34869-4_49

https://www.youtube.com/watch?v=9aYQvEYD4dc&ab_channel=CenterforIntelligentSystems%28formerlyMatYZ%29