Data Science Portfolio

Income Category Prediction

Table of contents

01

Business Problems

Business Problem

Perusahaan sedang menghadapi kesulitan dalam menetapkan harga yang tepat dan sesuai.

Oleh karena itu perlu dilakukakan analisis terkait pendapatan calon konsumen untuk memahami dengan akurat tingkat pendapatan mereka dan mengelompokkannya ke dalam 2 kelompok yang berbeda.

Informasi yang diperoleh dari analisis tersebut akan digunakan untuk mengoptimalkan strategi penetapan harga produk perusahaan, dengan harapan dapat meningkatkan pendapatan perusahaan, memenuhi kebutuhan calon konsumen, dan meningkatkan daya saing produk di pasar.

Purpose

01

Membangun model prediktif yang dapat memprediksi kategori pendapatan calon pelanggan.

Berdasarkan sejumlah variabel seperti umur, workclass, final weight, occupation, marital status, gender, dll sehingga dapat membantu perusahaan dalam memahami tingkat kemampuan beli pelanggan dan menyusun strategi harga yang lebih tepat.

02

Mencari pola antara karakteristik calon pelanggan dengan kategori income-nya.

03

Mengetahui pengaruh sejumlah variabel terhadap pendapatan calon pelanggan.

Sehingga memungkinkan perusahaan dalam menyusun strategi pemasaran dan harga yang lebih terfokus untuk setiap segmen.

02

Data Understanding

Dataset Information

Data ini merupakan data yang didapatkan dan dikumpulkan dari Biro Sensus di Amerika Serikat.

Berikut informasi dasar dari dataframe:

- Terdapat 48.841 baris data dan 15 kolom.
- DataFrame terdiri dari 6 kolom numerik dan 9 kolom kategorikal.
- Tidak ada value yang kosong di setiap kolom.

```
#info dasar dari dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48841 entries, 0 to 16279
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             48841 non-null  int64
 1   Workclass       48841 non-null  object
 2   Final Weight    48841 non-null  int64
 3   Education       48841 non-null  object
 4   EducationNum    48841 non-null  int64
 5   Marital Status  48841 non-null  object
 6   Occupation      48841 non-null  object
 7   Relationship    48841 non-null  object
 8   Race            48841 non-null  object
 9   Gender          48841 non-null  object
 10  Capital Gain    48841 non-null  int64
 11  capital loss    48841 non-null  int64
 12  Hours per Week  48841 non-null  int64
 13  Native Country  48841 non-null  object
 14  Income          48841 non-null  object
dtypes: int64(6), object(9)
memory usage: 6.0+ MB
```

Columns Explanation

Icome	Level gaji dari penduduk, terdiri dari 2 kategori yaitu gaji yang lebih dari 50,000 dollar dan gaji yang kurang dari atau sama dengan 50,000 dollar, keduanya ditulis dengan (>50K, <=50K).
Workclass	Tipe pekerjaan dari penduduk.
Marital-status	Status pernikahan.
Occupation	Bidang pekerjaan atau jabatan.
Relationship	Status hubungan.
Race	Ras dari penduduk.
Gender	Gender dari penduduk.
Capital-gain	Jumlah keuntungan modal (financial profit).
Capital-Loss	Jumlah kerugian modal.

Native-Country	Negara asal.
Age	Umur dari individu.
Hours-per-week	Jumlah jam kerja per-minggu.
Occupation	Bidang pekerjaan atau jabatan.
Final weight	Bobot nilai pada CPS yang diestimasi oleh Bureau Census Amerika Serikat.

Describe Dataset

1. Data Kategorikal

Memberikan gambaran awal mengenai distribusi atau nilai dalam setiap **kolom kategorikal.**

```
[ ] #Statistik dasar untuk semua kolom kategorikal
    df.describe(include='O')
```

	Workclass	Education	Marital Status	Occupation	Relationship	Race	Gender	Native Country	Income
count	48841	48841	48841	48841	48841	48841	48841	48841	48841
unique	9	16	7	15	6	5	2	42	4
top	Private	HS-grad	Married-civ-spouse	Prof-specialty	Husband	White	Male	United-States	<=50K
freq	33905	15784	22379	6172	19716	41762	32649	43831	24720

- **Count** : Jumlah entri dalam setiap kolom kategorikal.
- **Unique** : Menunjukkan jumlah nilai unik atau kategori yang berbeda.
- **Top** : Menunjukkan nilai atau kategori paling sering muncul.
- **Freq** : Menunjukkan seberapa sering nilai teratas muncul dalam setiap kolom.

2. Data Numerik

Memberikan gambaran awal mengenai distribusi atau nilai dalam setiap **kolom numerikal.**

```
#Statistik dasar untuk semua kolom numerik
df.describe()
```

	Age	Final Weight	EducationNum	Capital Gain	capital loss	Hours per Week
count	48841.000000	4.884100e+04	48841.000000	48841.000000	48841.000000	48841.000000
mean	38.643865	1.896634e+05	10.078152	1079.089720	87.504105	40.422391
std	13.710511	1.056050e+05	2.570961	7452.093748	403.008483	12.391571
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175490e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781420e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376460e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

03

Exploratory Data Analysis

Data Preparation

1. Handling Duplicates & Missing Values

- Terdapat 29 baris data yang terduplikasi pada dataframe.

```
#menghapus duplicate values
df = df.drop_duplicates()
```

```
df.duplicated().sum()
```

```
0
```

```
df
```

- Tidak terdapat null values pada setiap kolom di dataframe.

```
df.isna().sum()
```

```
Age               0
Workclass         0
Final Weight      0
Education         0
EducationNum      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
```

Data Cleaning

Terdapat beberapa kolom dengan value adalah ‘?’ (tanda tanya).

1. Kolom Workclass

```
[ ]  #Memeriksa jumlah dari masing-masing values #isi '?' ke "others"
     df['Workclass'].value_counts()
```

```
 Private             33878
 Self-emp-not-inc     3861
 Local-gov            3136
 ?                    2799
 State-gov            1981
 Self-emp-inc         1694
 Federal-gov          1432
 Without-pay            21
 Never-worked           10
Name: Workclass, dtype: int64
```

```
[ ]  #mengganti value "?" pada workclass menjadi "other"
     df['Workclass'] = df['Workclass'].replace(' ?', 'Other')
```

- Value “?” diganti ke “Other”.
- Hal ini dilakukan untuk memberikan kategorisasi umum yang mungkin lebih sesuai untuk data yang tidak dapat diidentifikasi atau diklasifikasikan secara spesifik.

Data Cleaning

Terdapat beberapa kolom dengan value adalah ‘?’ (tanda tanya).

2. Kolom Occupation

```
#Memeriksa jumlah dari masing-masing values
df['Occupation'].value_counts()
```

```
Prof-specialty       6167
Craft-repair         6107
Exec-managerial      6084
Adm-clerical         5608
Sales                5504
Other-service        4919
Machine-op-inspct    3018
?                    2809
Transport-moving     2355
Handlers-cleaners    2071
Farming-fishing      1487
Tech-support         1445
Protective-serv       983
Priv-house-serv       240
Armed-Forces           15
Name: Occupation, dtype: int64
```

```
#mengganti value "?" pada occupation menjadi "other"
df['Occupation'] = df['Occupation'].replace(' ?', ' Other-service')
```

Value “?” dimasukkan ke “Other-service”.

Data Cleaning

Menyederhanakan values pada beberapa kolom

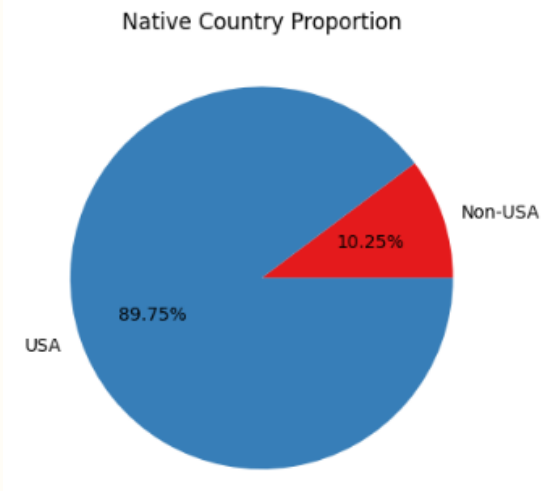
3. Kolom Native Country

```
# Simplify Native Country
df["Native Country"] = np.where(df["Native Country"]==' United-States', 'USA', 'Non-USA')

Native_Country = df.groupby(["Native Country"]).size()

plt.title('Native Country Proportion')
plt.pie(Native_Country, labels=Native_Country.index, autopct=lambda p: f'{p:.2f}%')
plt.show();
```

- Karena 90% penduduk bernegara asal United States (USA) dan values yang lain (negara selain dari USA) berjumlah sangat sedikit.
- Maka dilakukan penggabungkan nilai-nilai minoritas menjadi satu kategori (non-USA).



Data Cleaning

Menyederhanakan values pada beberapa kolom

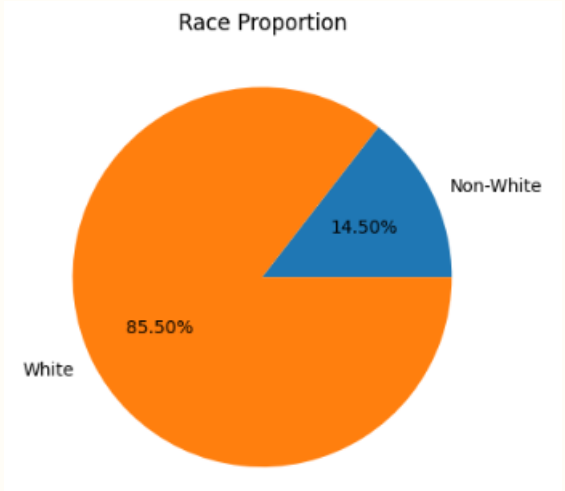
3. Kolom Race

```
[ ]  # Simplify Race
     df["Race"] = np.where(df["Race"]=='White', 'White', 'Non-White')

     Race = df.groupby(["Race"]).size()

     plt.title('Race Proportion')
     plt.pie(Race, labels=Race.index, autopct=lambda p: f'{p:.2f}%')
     plt.show();
```

- Karena 85% penduduk rasnya adalah white dan penduduk dengan ras yang lain berjumlah sangat sedikit, maka values yang lain akan digabungkan menjadi kategori (Non-White).



Data Cleaning

5. Kolom Marital Status

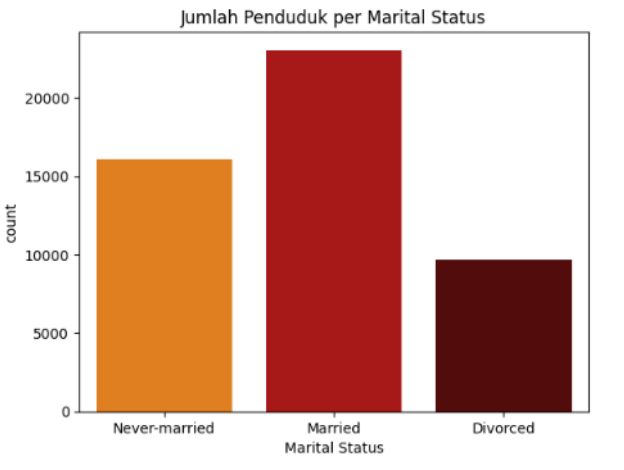
```
[50] df['Marital Status'].unique()

     array(['Never-married', 'Married-civ-spouse', 'Divorced',
            'Married-spouse-absent', 'Separated', 'Married-AF-spouse',
            'Widowed'], dtype=object)

[51] df["Marital Status"] = df["Marital Status"].replace(['Divorced','Separated', 'Widowed'], "Divorced")

[52] df["Marital Status"] = df["Marital Status"].replace(['Married-civ-spouse','Married-spouse-absent', 'Married-AF-spouse'], "Married")
```

- Menyederhanakan values pada beberapa kolom.
- Untuk memudahkan pengelompokan, maka beberapa values pada kolom Marital Status perlu di gabungkan.



Data Cleaning

Menghapus space di setiap kata awal data kategorikal

6. Menghapus Space

```
Marital Status - (7) : [' Never-married' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed'] ,
Occupation - (15) : [' Adm-clerical' ' Exec-managerial' ' Handlers-cleaners' ' Prof-specialty'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support' ' ?'
 ' Protective-serv' ' Armed-Forces' ' Priv-house-serv'] ,
```

Sebelum

Begitupun dengan kolom Occupation

```
[33] #menggunakan funcion str.strip.() untuk menghapus spasi
     for column in df:
       if df[column].dtype == 'object':
         df[column] = df[column].str.strip()
```

Code

```
Marital Status - (7) : ['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-absent'
 'Separated' 'Married-AF-spouse' 'Widowed'] ,
Occupation - (14) : ['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-specialty'
 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' 'Protective-serv'
 'Armed-Forces' 'Priv-house-serv'] ,
```

Sesudah

Data Cleaning

Memperbaiki values pada kolom Income

7. Kolom Income

Value yang sama

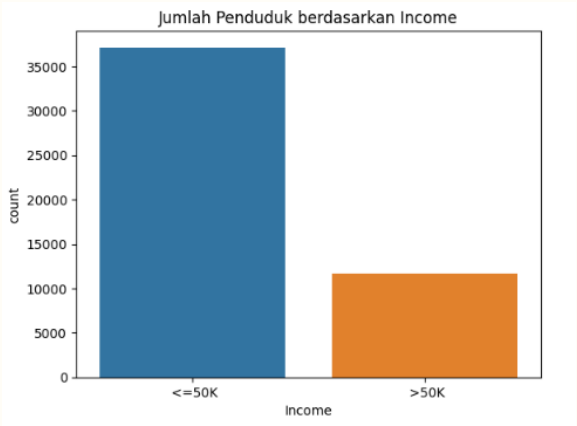
```
Income - (4) : ['<=50K' '>50K' '<=50K.' '>50K.'] ,
```

Value yang sama

Sebelum

```
df.replace({'Income':{" >50K.":" >50K" ,  " <=50K.":" <=50K"}},inplace=True)
```

Code

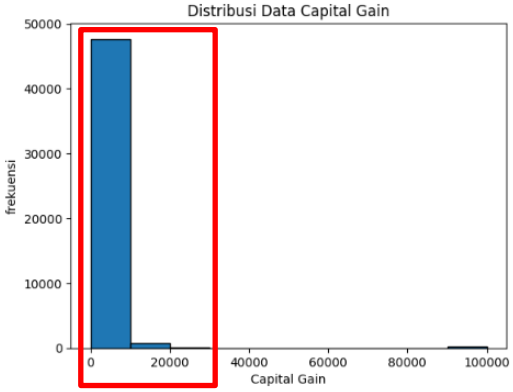


Sesudah

Data Cleaning

Drop kolom numerik yang terlalu imbalance

8. Kolom Capital Gain



Sebelum

```
[36] #Memeriksa persentase dari values 0 di capital gain
     len1 = round((len(df[df['Capital Gain']== 0])/len(df)*100),2)
     print(len1,"%")

     91.73 %
```

Code

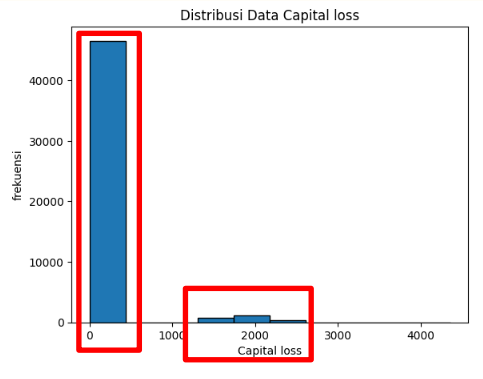
```
[37] #menghapus kolom Capital Gain dari Dataframe
     df.drop(columns='Capital Gain', inplace=True)
```

- Kolom numerik yang sangat tidak seimbang dapat mempengaruhi performa model atau analisis, sehingga dilakukan penghapusan pada kolom ini.

Data Cleaning

Drop kolom numerik yang terlalu imbalance

9. Kolom Capital Loss




```
#Memeriksa persentase dari values 0 di capital loss
len2 = round((len(df[df['capital loss']== 0])/len(df)*100),2)
print(len2,"%")
```

```
95.32 %
```

Sebelum

```
[41] #menghapus kolom Capital Loss dari Dataframe
     df.drop(columns='capital loss', inplace=True)
```

Code

Data Cleaning

Drop salah satu kolom karena identik

10. Education vs EducationNum

Variable	VIF
EducationNum	inf
Education	inf

- Nilai "inf" pada kolom VIF menunjukkan bahwa ada indikasi kuat terjadinya masalah multikolineritas, dan juga menunjukkan bahwa kemungkinan variabel tersebut memiliki korelasi yang sangat tinggi dengan variabel lain. Artinya variabel Education dapat sepenuhnya dijelaskan oleh variabel EducationNum.
- Untuk menghindari masalah tersebut maka kolom Education akan dihapus dan EducationNum akan digunakan.

```
#Menghapus kolom Education
df.drop(columns='Education', inplace=True)
```

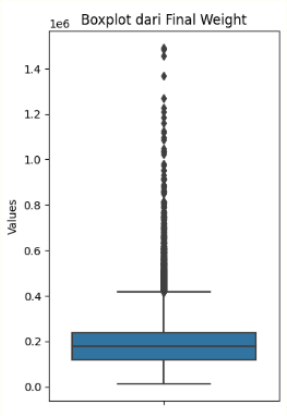
Code

Data Cleaning

Mengatasi outlier pada kolom numerik

11. Kolom Final Weight

Sebelum:



Menggunakan metode cap outlier

```
# Calculate the IQR
q1 = np.percentile(df["Final Weight"], 25)
q3 = np.percentile(df["Final Weight"], 75)
iqr = q3 - q1
# Find outliers
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = df["Final Weight"][(df["Final Weight"] < lower_bound) | (df["Final Weight"] > upper_bound)].
```

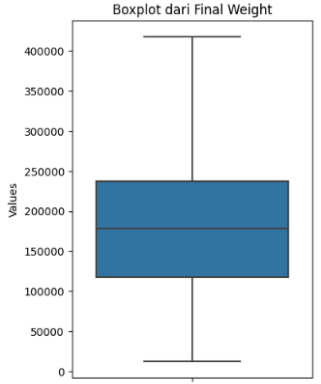
```
[69] outliers.count()
```

```
1453
```

Terdapat 1453 baris pada kolom final weight yang tergolong outlier

```
[70] #Mengatasi outlier dengan menggunakan cap method
     # Cap outliers using the IQR method
     df['Final Weight'] = np.clip(df['Final Weight'], lower_bound, upper_bound)
```

Sesudah:



Data Cleaning

DataFrame setelah proses cleaning

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48812 entries, 0 to 16279
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Age             48812 non-null  int64 
 1   Workclass       48812 non-null  object
 2   Final Weight    48812 non-null  int64 
 3   EducationNum    48812 non-null  int64 
 4   Marital Status  48812 non-null  object
 5   Occupation      48812 non-null  object
 6   Relationship    48812 non-null  object
 7   Race            48812 non-null  object
 8   Gender          48812 non-null  object
 9   Hours per Week  48812 non-null  int64 
 10  Native Country  48812 non-null  object
 11  Income          48812 non-null  object
dtypes: int64(4), object(8)
memory usage: 4.8+ MB
```

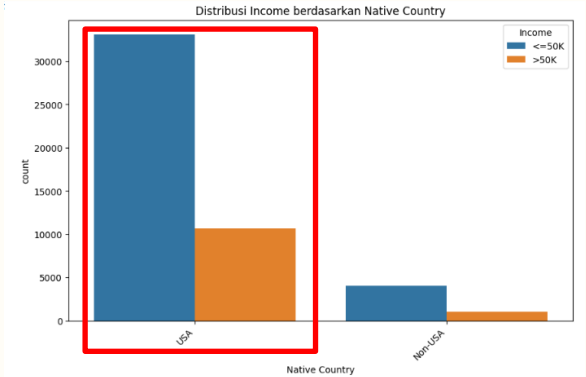
	Age	Workclass	Final Weight	EducationNum	Marital Status	Occupation	Relationship	Race	Gender	Hours per Week	Native Country	Income
0	39	State-gov	77516	13	Never-married	Adm-clerical	Not-in-family	White	Male	40	USA	<=50K
1	50	Self-emp-not-inc	83311	13	Married	Exec-managerial	Husband	White	Male	13	USA	<=50K
2	38	Private	215646	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	40	USA	<=50K
3	53	Private	234721	7	Married	Handlers-cleaners	Husband	Non-White	Male	40	USA	<=50K
4	28	Private	338409	13	Married	Prof-specialty	Wife	Non-White	Female	40	Non-USA	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...
16275	39	Private	215419	13	Divorced	Prof-specialty	Not-in-family	White	Female	36	USA	<=50K
16276	64	Other	321403	9	Divorced	Other	Other-relative	Non-White	Male	40	USA	<=50K
16277	38	Private	374983	13	Married	Prof-specialty	Husband	White	Male	50	USA	<=50K
16278	44	Private	83891	13	Divorced	Adm-clerical	Own-child	Non-White	Male	40	USA	<=50K
16279	35	Self-emp-inc	182148	13	Married	Exec-managerial	Husband	White	Male	60	USA	>50K

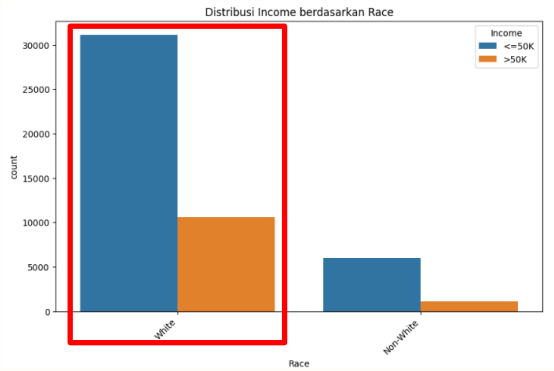
48812 rows × 12 columns

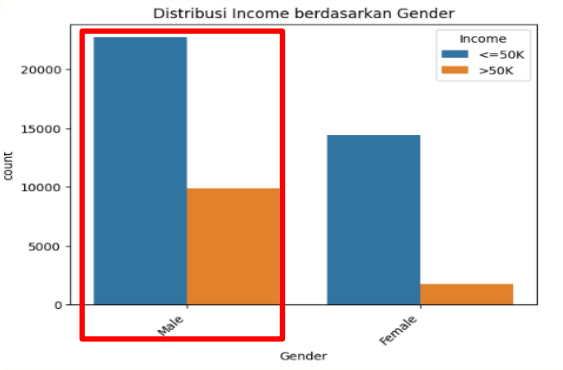
- Data awal sebanyak 48.841 baris data dan 15 kolom, setelah dilakukan proses cleaning data menjadi 48.812 baris data dan 12 kolom.
- Penghapusan kolom Education, Capital Gain, dan Capital Loss.
- Penghapusan duplicate value sebanyak 29 baris data.

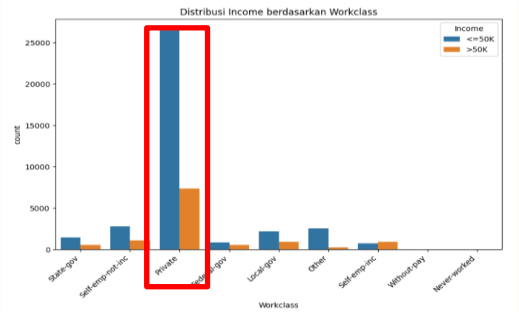
Data Visualization

1. Distribusi Income berdasarkan Native Country, Race, Gender dan Workclass





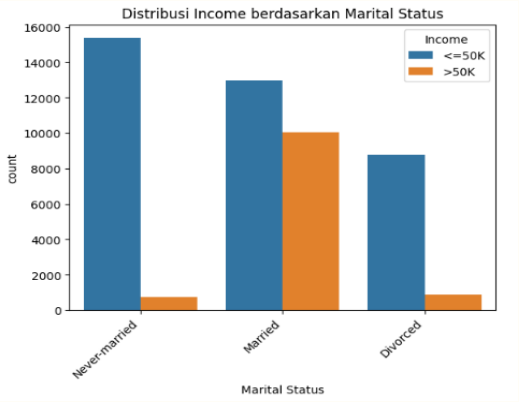


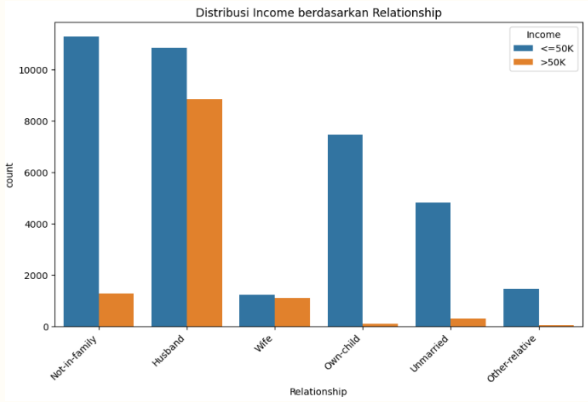


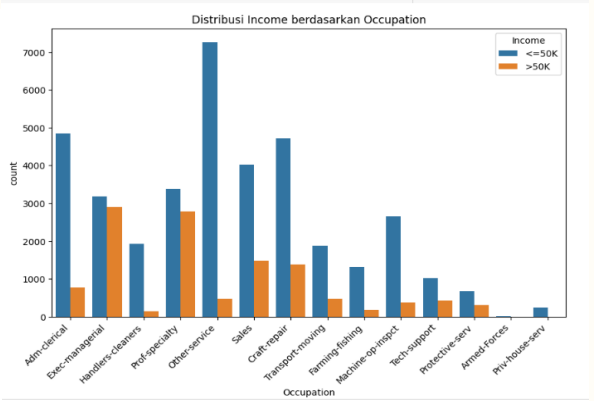
4 fitur ini memiliki pola yang sama, dimana distribusi income <=50 K dan >50K didominasi oleh salah satu value pada setiap kolom.

Data Visualization

2. Distribusi Income berdasarkan Marital Status, Relationship dan Occupation



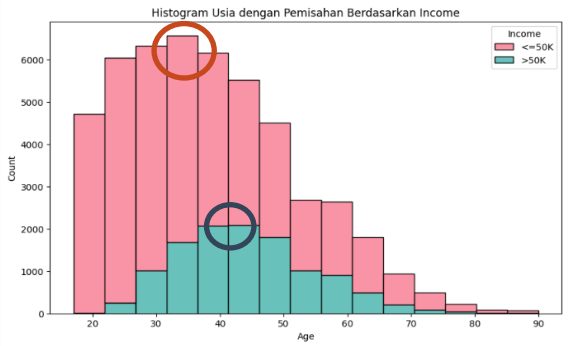


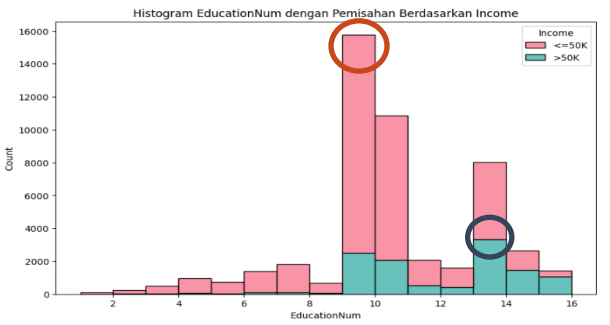


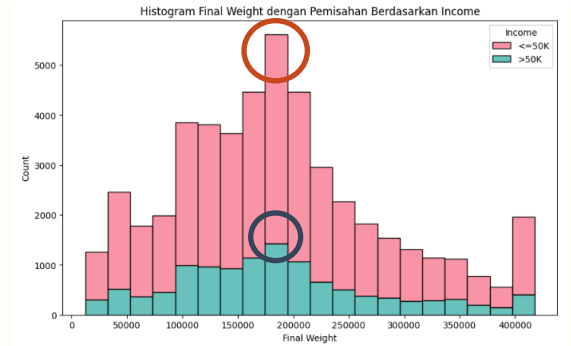
3 fitur ini memiliki pola yang sama, dimana distribusi income <=50 K dan >50K didominasi oleh dua values yang berbeda.

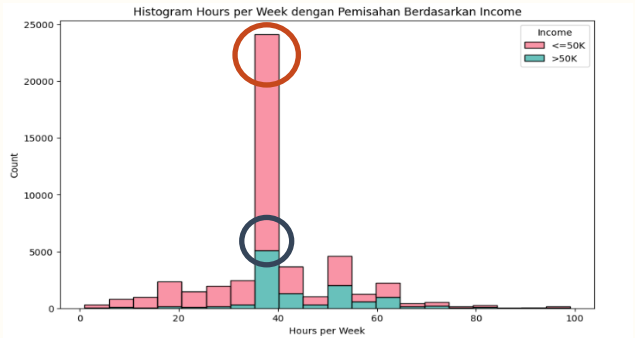
Data Visualization

3. Distribusi Age, EducationNum, Final Weight dan Hours per Week berdasarkan Income









Titik rata-rata untuk tingkat income <=50K

Titik rata-rata untuk tingkat income >50K

04

Data Pre-Processing

Data Encoding

1. Label Encoding

Label Encoder dilakukan pada variabel target “Income” yang akan mengubah data :

- Values <=50K diberikan label 0
- Values >50K diberikan label 1

```python
#income di ubah ke 0 dan 1
from sklearn.preprocessing import LabelEncoder

def label_encoder(dataframe, binary_col):
    encoders = {}  # untuk menyimpan objek encoder untuk kolom-kolom yang di-encode
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.fit_transform(dataframe[binary_col])
    encoders[binary_col] = labelencoder  # Save the encoder for future use
    return dataframe, encoders

# Example usage:
# Replace 'df3' with your actual DataFrame and 'Income' with the binary column you want to encode.
df3, encoders = label_encoder(df3, 'Income')

# Display the transformed DataFrame
print(df3)

# To use the encoder later:
# encoded_values = encoders['Income'].transform(some_values)
```

```
       Income
0           0
1           0
2           0
3           0
4           0
...       ...
16275       0
16276       0
16277       0
16278       0
16279       1

[48812 rows x 12 columns]
```

2. One-Hot Encoding

Terdapat 7 fitur yang diberikan label menggunakan metode One-Hot Encoding, yaitu: Workclass, Marital Status, Occupation, Relationship, Race, Gender dan Native Country.

Sebelum:

df3

	Age	Workclass	Final Weight	EducationNum	Marital Status	Occupation	Relationship	Race	Gender	Hours per Week	Native Country	Income
0	39	State-gov	77516	13	Never-married	Adm-clerical	Not-in-family	White	Male	40	USA	0
1	50	Self-emp-not-inc	83311	13	Married	Exec-managerial	Husband	White	Male	13	USA	0
2	38	Private	215646	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	40	USA	0
3	53	Private	234721	7	Married	Handlers-cleaners	Husband	Non-White	Male	40	USA	0
4	28	Private	338409	13	Married	Prof-specialty	Wife	Non-White	Female	40	Non-USA	0
...	...	...	...	...	...	...	...	...	...	...	...	...
16275	39	Private	215419	13	Divorced	Prof-specialty	Not-in-family	White	Female	36	USA	0
16276	64	Other	321403	9	Divorced	Other-service	Other-relative	Non-White	Male	40	USA	0
16277	38	Private	374983	13	Married	Prof-specialty	Husband	White	Male	50	USA	0
16278	44	Private	83891	13	Divorced	Adm-clerical	Own-child	Non-White	Male	40	USA	0
16279	35	Self-emp-inc	182148	13	Married	Exec-managerial	Husband	White	Male	60	USA	1

48812 rows × 12 columns

- Dilakukan drop_first=True menunjukkan bahwa penghapusan kolom pertama hasil one-hot encoding untuk menghindari multicollinearity.
- Multicollinearity adalah fenomena dimana dua atau lebih variabel bebas dalam model regresi memiliki tingkat korelasi yang tinggi satu sama lain.

Sesudah:

```
[ ] df3 = pd.get_dummies(df3, drop_first=True) # one hot encoding untuk fitur kategorikal di dataset
    print(df3.shape)

    (48812, 36)

[ ] df3
```

	Age	Final Weight	EducationNum	Hours per Week	Income	Workclass_Local-gov	Workclass_Never-worked	Workclass_Other	Workclass_Private	Workclass_Self-emp-inc	Workclass_Self-emp-not-inc	Workclass_St
0	39	77516	13	40	0	0	0	0	0	0	0	
1	50	83311	13	13	0	0	0	0	0	0	1	
2	38	215646	9	40	0	0	0	0	1	0	0	
3	53	234721	7	40	0	0	0	0	1	0	0	
4	28	338409	13	40	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
16275	39	215419	13	36	0	0	0	0	1	0	0	
16276	64	321403	9	40	0	0	0	1	0	0	0	
16277	38	374983	13	50	0	0	0	0	1	0	0	
16278	44	83891	13	40	0	0	0	0	1	0	0	
16279	35	182148	13	60	1	0	0	0	0	1	0	

48812 rows × 36 columns

Splitting Dataset

Dataset

- **Variabel target (Y)**
  - Income
- **Variabel prediktor (x)**
  - Selain variabel income

```
[ ] X_train, X_test, y_train, y_test = train_test_split(df4, target,
                                                        stratify=target, # supaya ratio value di data training dan test sama
                                                        test_size = 0.2, #use 0.1 if data is huge.
                                                        random_state = 0)

    #to resolve any class imbalance - use stratify parameter.

    print("Dimensi X_train dataset: ", X_train.shape)
    print("Dimensi y_train dataset: ", y_train.shape)
    print("Dimensi X_test dataset: ", X_test.shape)
    print("Dimensi y_test dataset: ", y_test.shape)
```

```
Dimensi X_train dataset:  (39049, 35)
Dimensi y_train dataset:  (39049,)
Dimensi X_test dataset:  (9763, 35)
Dimensi y_test dataset:  (9763,)
```

- Dataset dibagi menjadi Data Train dan Data Test.
- Dengan rasio yang digunakan 80:20 atau Data Train (80%) dan Data Test (20%).

X_test

	Age	Final Weight	EducationNum	Hours per Week	Workclass_Local-gov	Workclass_Never-worked	Workclass_Other	Workclass_Private	Workclass_Self-emp-inc	Workclass_Self-emp-not-inc	Work
6286	24	138938	10	40	0	0	0	1	0	0	
11843	35	261646	9	40	0	0	0	1	0	0	
7719	23	239539	10	40	0	0	0	1	0	0	
25713	32	183304	11	99	0	0	0	1	0	0	
19370	55	101468	9	40	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	
718	35	179579	14	48	0	0	0	1	0	0	
5432	29	176137	9	40	0	0	0	1	0	0	
394	37	79586	9	60	0	0	0	1	0	0	
7065	24	161092	13	40	0	0	0	1	0	0	
29004	45	48495	13	42	0	0	0	1	0	0	

9763 rows × 35 columns

X_test

X_train

X_train

	Age	Final Weight	EducationNum	Hours per Week	Workclass_Local-gov	Workclass_Never-worked	Workclass_Other	Workclass_Private	Workclass_Self-emp-inc	Workclass_Self-emp-not-inc	Work
29265	36	148581	9	55	0	0	0	1	0	0	
3249	40	227823	12	70	0	0	0	1	0	0	
31055	54	222882	9	45	0	0	0	1	0	0	
14883	48	193775	13	38	0	0	0	1	0	0	
3846	29	57596	10	40	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	
20411	48	246891	15	60	0	0	0	0	0	1	
15606	49	243190	11	40	0	0	0	1	0	0	
2274	48	119565	9	40	0	0	0	1	0	0	
12802	17	117549	6	12	0	0	0	1	0	0	
6926	53	27166	9	40	0	0	0	1	0	0	

39049 rows × 35 columns

Feature Scaling

Fitur-fitur di-scaling menggunakan robust scaler karena lebih tahan terhadap outlier dan cocok untuk data yang tidak terdistribusi normal.

```
[ ] from sklearn.preprocessing import RobustScaler
    sc_X = RobustScaler()
    X_train2 = pd.DataFrame(sc_X.fit_transform(X_train))
    X_train2.columns = X_train.columns.values
    X_train2.index = X_train.index.values
    X_train = X_train2

    X_test2 = pd.DataFrame(sc_X.transform(X_test))
    X_test2.columns = X_test.columns.values
    X_test2.index = X_test.index.values
    X_test = X_test2
```

X_train

	Age	Final Weight	EducationNum	Hours per Week	Workclass_Local-gov	Workclass_Never-worked	Workclass_Other	Workclass_Private	Workclass_Self-emp-inc	Workclass_Self-emp-not-inc	W
29265	-0.05	-0.247711	-0.333333	3.0	0.0	0.0	0.0	0.0	0.0	0.0	
3249	0.15	0.410785	0.666667	6.0	0.0	0.0	0.0	0.0	0.0	0.0	
31055	0.85	0.369725	-0.333333	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
14883	0.55	0.127848	1.000000	-0.4	0.0	0.0	0.0	0.0	0.0	0.0	
3846	-0.40	-1.003789	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
20411	0.55	0.569238									
15606	0.60	0.538483									
2274	0.55	-0.488831									
12802	-1.00	-0.505584									
6926	0.80	-1.256660									

39049 rows × 35 columns

X_test

	Age	Final Weight	EducationNum	Hours per Week	Workclass_Local-gov	Workclass_Never-worked	Workclass_Other	Workclass_Private	Workclass_Self-emp-inc	Workclass_Self-emp-not-inc	W
6286	-0.65	-0.327843	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
11843	-0.10	0.691851	-0.333333	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
7719	-0.70	0.508144	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
25713	-0.25	0.040835	0.333333	11.8	0.0	0.0	0.0	0.0	0.0	0.0	
19370	0.90	-0.639216	-0.333333	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
718	-0.10	0.009881	1.333333	1.6	0.0	0.0	0.0	0.0	0.0	0.0	
5432	-0.40	-0.018722	-0.333333	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
394	0.00	-0.821054	-0.333333	4.0	0.0	0.0	0.0	0.0	0.0	0.0	
7065	-0.65	-0.143745	1.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
29004	0.40	-1.079418	1.000000	0.4	0.0	0.0	0.0	0.0	0.0	0.0	

9763 rows × 35 columns

05

Data Modeling

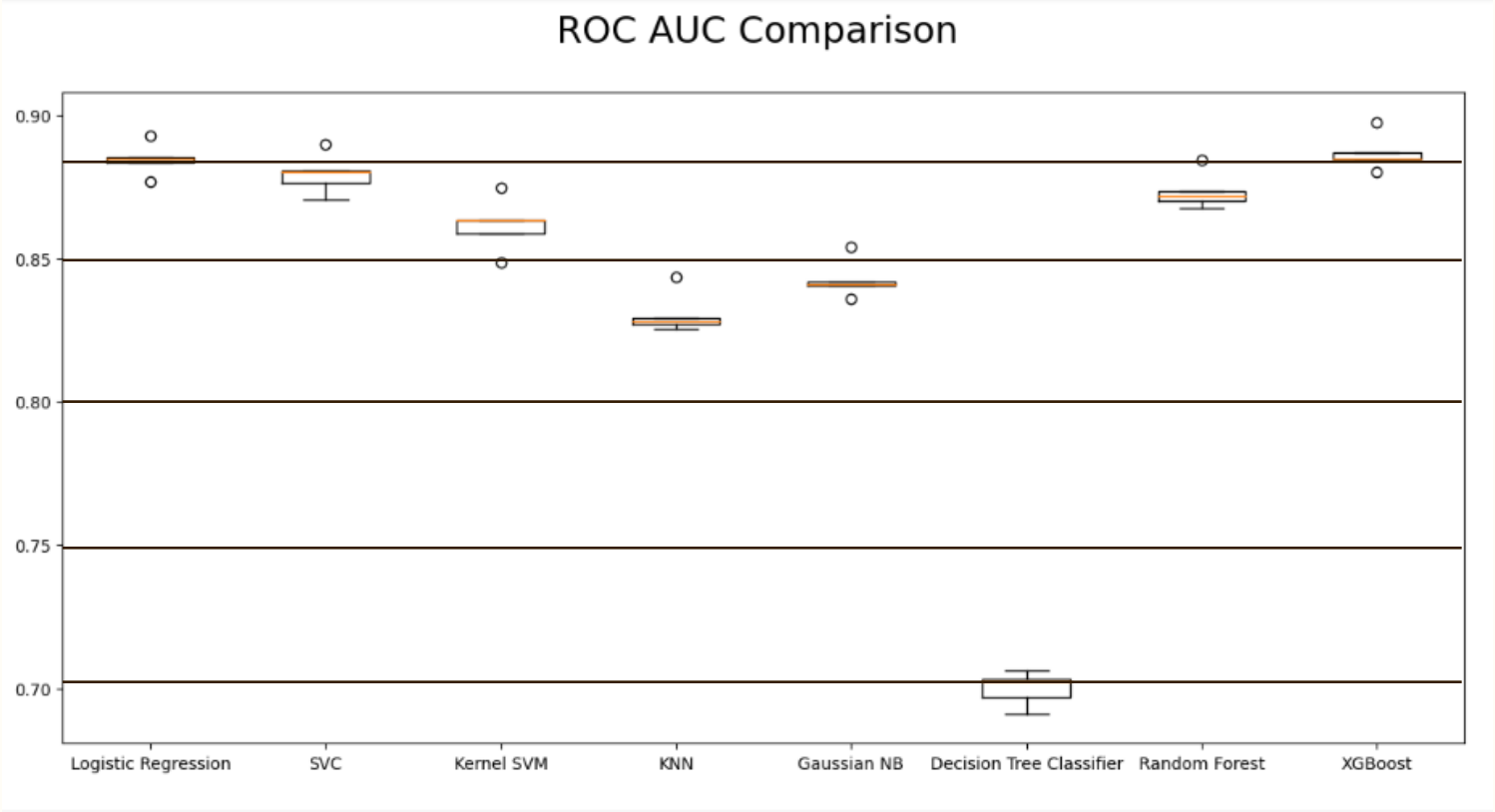
Model Selection

Pemilihan Model menggunakan metode K-fold Cross Validation dengan split sebanyak 5 kali.

	Algorithm	ROC AUC Mean	ROC AUC STD	Accuracy Mean	Accuracy STD
7	XGBoost	88.68	0.58	83.40	0.49
0	Logistic Regression	88.47	0.51	78.77	0.34
1	SVC	87.97	0.63	83.06	0.48
6	Random Forest	87.34	0.57	82.71	0.44
2	Kernel SVM	86.87	0.84	83.69	0.40
4	Gaussian NB	84.27	0.60	65.81	1.15
3	KNN	83.65	0.82	81.59	0.53
5	Decision Tree Classifier	69.99	0.50	77.73	0.37

- Menggunakan ROC_AUC Mean sebagai patokan karena ROC_AUC Mean adalah rata-rata dari ROC_AUC.
- Jadi ROC_AUC Mean lebih akurat daripada ROC_AUC.

Distribusi nilai ROC AUC dengan K-fold Cross Validation.



Model Evaluation

Classification Metrics

Tanpa Hyperparameter

Model	Accuracy	Precision	Recall	F1 Score	F2 Score	ROC-AUC
XGBoost	0.84	0.70	0.59	0.64	0.61	**0.75**

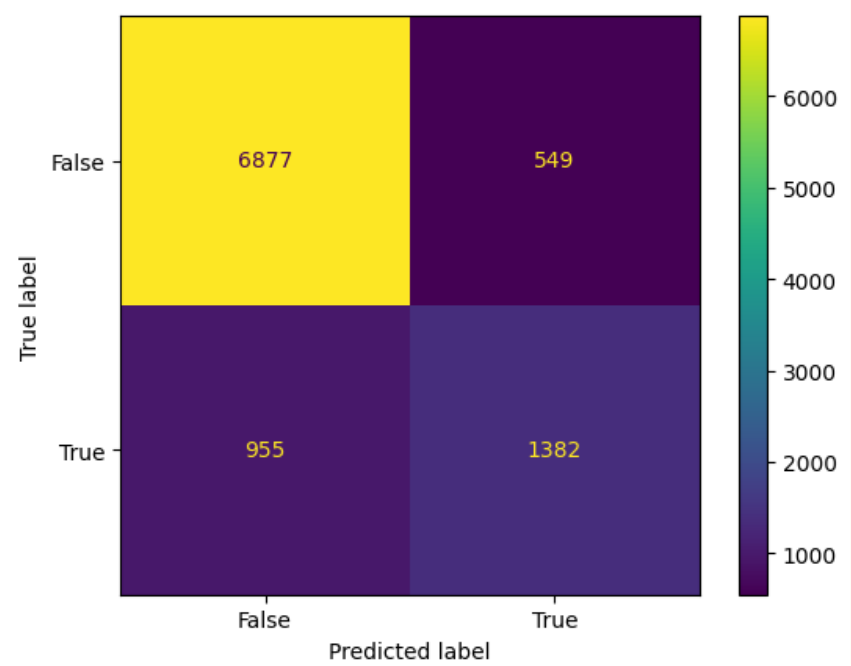
Dengan Hyperparameter

Model	Accuracy	Precision	Recall	F1 Score	F2 Score	ROC-AUC
XGBoost	0.85	0.72	0.59	0.65	0.61	**0.76**

- Terjadi sedikit peningkatan pada model XGBoost setelah dilakukan hyperparameter tuning.
- Selain itu, faktor datasetnya balanced atau imbalanced juga diperhitungkan. Jika data balanced accuracy akan menjadi utama, jika data imbalanced maka ROC_AUC diutamakan.
- Karena data target imbalance, maka metrik evaluasi yang akan digunakan nanti adalah ROC/AUC.

Model Evaluation

Confusion Metrics



Prediksi income >50K yang sebenarnya >50K adalah 1382 **(True Positive).**

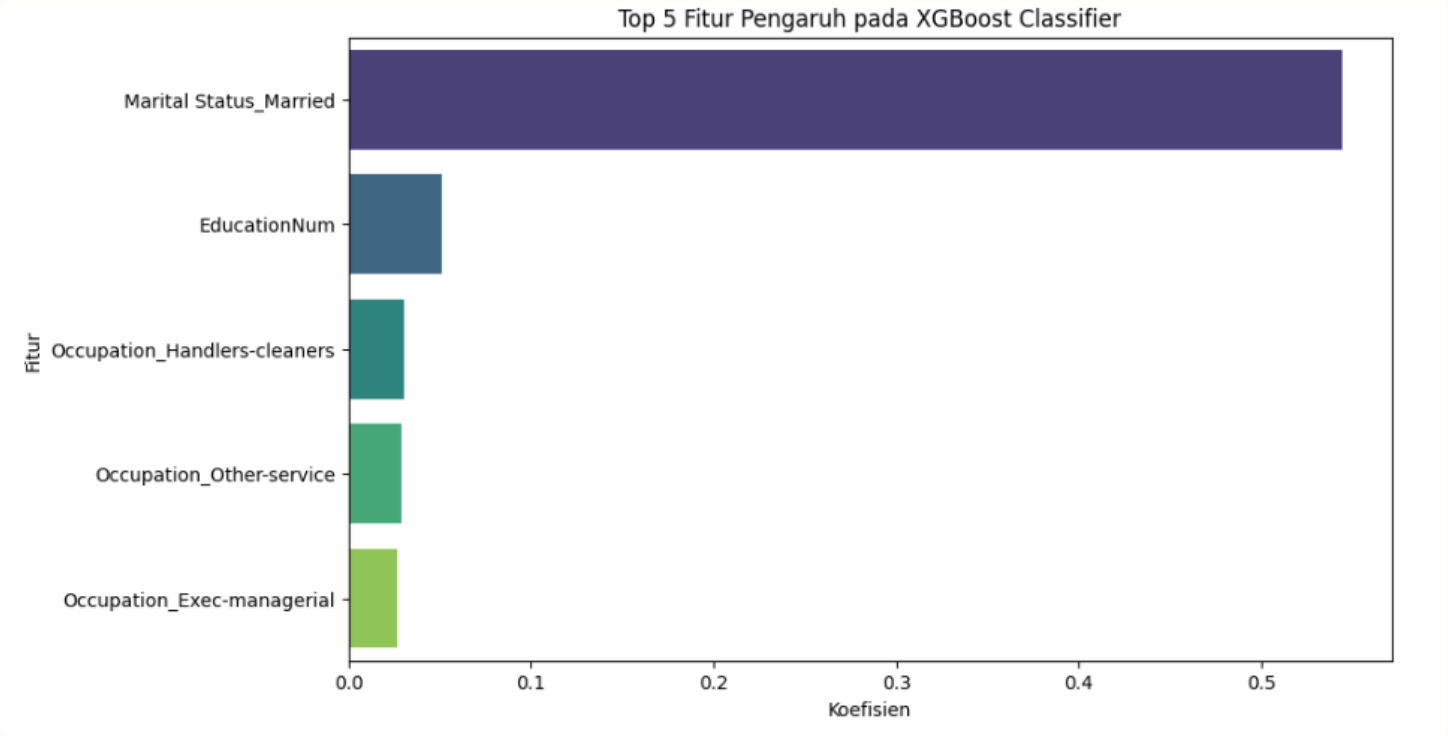
Prediksi income <=50K yang sebenarnya >50K adalah 955 **(False Negative).**

Prediksi income <=50K yang sebenarnya <=50K adalah 6877 **(True Negative).**

Prediksi income >50K yang sebenarnya <=50K adalah 549 **(False Positive).**

Features Importants

5 fitur dengan pengaruh terbesar terhadap kategori target



06

Model Deployment

Tahap Model Deployment

- Dalam proses model deployment, model machine learning yang sudah di-training dan dievaluasi yaitu XGBoost disimpan dalam file dengan format pickle.
- File pickle diimpor ke dalam code deployment yang kemudian diedit dalam code editor VS Code.
- Code deployment dan file kebutuhan deploy lainnya diupload ke repository Github yang sudah dikoneksikan ke framework Streamlit.
- Dalam tampilan UI dengan framework Streamlit, dibuat dua menu yaitu halaman “Home” yang menampilkan deskripsi dari app tersebut dan link sumber data yang digunakan, dan halaman “Machine Learning” yang menampilkan attribute info, form untuk input data baru yang akan diprediksi, dan hasil prediksinya.

Input Your Data

Age

35

Workclass

Private

Final Weight

283745

Education

Bachelors

Education Number

16

Marital Status

Married-spouse-absent

40

Native Country

Germany

Submit

Your Selected Options :

ıcationNum	Marital Status	Occupation	Relationship	Race	Gender	Capital Gain	capital loss	Hours per Week	Native Country
16	Married-spouse-absent	Exec-managerial	Husband	White	Male	1000	0	40	Germany

Prediction Result

Hasil Prediksi Income : > 50K

Gaji lebih dari 50,000 dollar

07

Conclusions

Conclusions

01

Berdasarkan proses analisis mulai dari data cleaning sampai dengan modeling, didapatkan hasil bahwa untuk memprediksi tingkat income dari calon pelanggan, model yang paling bagus yaitu **XGBoost** dengan nilai **ROC/AUC** sebesar **0,75** dan **ROC/AUC mean** sebesar **0,88**

02

Berdasarkan proses EDA, value **Marital Status "Never-married"**, **Relationship "Not-in-family"**, dan **Occupation "Other-service"** memiliki pola dengan variabel income yang didominasi dengan income <=50K.

03

Terdapat 5 karakteristik teratas yang mempengaruhi prediksi tingkat income, yaitu **Marital Status "Married", EducationNum, Occupation_"Handlers-cleaners", Occupation_"Other-service", Occupation_"Exec-managerial".**

Link Address

Data Source : Census Income Dataset

Google Colab : Income Category – Google Colab

Github : https://github.com/inneandarinii/Data-Science-Project-8

Streamlit : https://finalprojectincomef4.streamlit.app/

Inne Andarini Herdianti S. Si

Data Science Enthusiast

A bachelor of Science degree in Physics was obtained from the Bandung Institute of Technology. I'm eager to dive into the world of data science. Please check out some of the projects I've worked on my GitHub or LinkedIn. I'm excited about the opportunity to bring my skills and enthusiasm for Data Science!

Contact:

https://www.linkedin.com/in/inneandarini/

inneandarinii@gmail.com

https://github.com/inneandarinii

Explore my portfolio dashboard at

datascienceportfol.io/inneandarini

Thanks!