Census Income EDA

Prepared By:

Bankole Ayoade

GitHub: https://github.com/acaaattunde2012

LinkedIn: https://www.linkedin.com/in/bankole-ayoade-fca-acti-cipfa-161406a7/

About Census Income Dataset

The dataset comprises 32561 rows and 15 coulmns as follows:

- income_class

  > 50K, <=50K.

- age: continuous.
- workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- fnlwgt: continuous.
- education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, * 10th, Doctorate, 5th-6th, Preschool.
- education-num: continuous.
- marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex: Female, Male.
- capital-gain: continuous.
- capital-loss: continuous.
- hours-per-week: continuous.
- native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Importing Dependencies

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"

warnings.filterwarnings("ignore")

%matplotlib inline
```

Data Ingestion:

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'
```

```
#income_df=pd.read_csv(url, header=None)
income_df=pd.read_csv("adultdata.csv", header=None)
```

```
income_df.head(5)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-	Handlers-							United-	

```
# Insert column names
col_name = ["age", "workclass","fnlwgt","education","education-num","marital_status","occupation","relationship","race",
            "sex","capital_gain","capital_loss","hour_per_week","native_country","income_class"]
income_df.columns = col_name
income_df.head(5)
```

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0

```
# Make a copy of the dataset
income = income_df.copy()
```

Data Profiling

```
# Check the first 5 rows
income.head(5)
```

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0

```
# Check the last 5 rows
income.tail(5)
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital_gain	capital_loss
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0

```
# Insert column names
col_name = ["age", "workclass","fnlwgt","education","education-num","marital_status","occupation","relationship","race",
            "sex","capital_gain","capital_loss","hour_per_week","native_country","income_class"]
income.columns = col_name
income.head(5)
```

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0

```
# Show the no of rows and columns
income.shape
```

```
(32561, 15)
```

```
# Show more info about the dataset
income.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64 
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital_gain    32561 non-null  int64 
 11  capital_loss    32561 non-null  int64 
 12  hour_per_week   32561 non-null  int64 
 13  native_country  32561 non-null  object
 14  income_class    32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
# Checking missing values
income.isnull().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hour_per_week     0
native_country    0
```

```
income_class      0
dtype: int64
```

```
# Checking missing values
income.isna().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hour_per_week     0
native_country    0
income_class      0
dtype: int64
```

```
# Checking for duplicates
income[income.duplicated()]
```

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital_gain	capita
4881	25	Private	308144	Bachelors	13	Never-married	Craft-repair	Not-in-family	White	Male	0	
5104	90	Private	52386	Some-college	10	Never-married	Other-service	Not-in-family	Asian-Pac-Islander	Male	0	
9171	21	Private	250051	Some-	10	Never-married	Prof-	Own-child	White	Female	0	

```
income.drop_duplicates(keep='first', inplace=True)
income[income.duplicated()]
```

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hou
							fishing						

```
# Checking for unique values for each column
for col in income.columns:
    print(f"==========================================={col}===============================================================")
    print(income[col].unique())
    print("=============================================================================================================")

```

```
===========================================age================================================================
[39 50 38 53 28 37 49 52 31 42 30 23 32 40 34 25 43 54 35 59 56 19 20 45
 22 48 21 24 57 44 41 29 18 47 46 36 79 27 67 33 76 17 55 61 70 64 71 68
 66 51 58 26 60 90 75 65 77 62 63 80 72 74 69 73 81 78 88 82 83 84 85 86
 87]
=============================================================================================================
===========================================workclass==========================================================
[' State-gov' ' Self-emp-not-inc' ' Private' ' Federal-gov' ' Local-gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked']
=============================================================================================================
===========================================fnlwgt=============================================================
[ 77516  83311 215646 ...  34066  84661 257302]
=============================================================================================================
===========================================education==========================================================
[' Bachelors' ' HS-grad' ' 11th' ' Masters' ' 9th' ' Some-college'
 ' Assoc-acdm' ' Assoc-voc' ' 7th-8th' ' Doctorate' ' Prof-school'
 ' 5th-6th' ' 10th' ' 1st-4th' ' Preschool' ' 12th']
=============================================================================================================
===========================================education-num======================================================
[13  9  7 14  5 10 12 11  4 16 15  3  6  2  1  8]
=============================================================================================================
===========================================marital_status=====================================================
[' Never-married' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed']
=============================================================================================================
===========================================occupation=========================================================
[' 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']
=============================================================================================================
===========================================relationship=======================================================
[' Not-in-family' ' Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']
=============================================================================================================
===========================================race===============================================================
[' White' ' Black' ' Asian-Pac-Islander' ' Amer-Indian-Eskimo' ' Other']
=============================================================================================================
===========================================sex================================================================
[' Male' ' Female']
=============================================================================================================
===========================================capital_gain=======================================================
[ 2174     0 14084  5178  5013  2407 14344 15024  7688 34095  4064  4386
  7298  1409  3674  1055  3464  2050  2176   594 20051  6849  4101  1111
  8614  3411  2597 25236  4650  9386  2463  3103 10605  2964  3325  2580
  3471  4865 99999  6514  1471  2329  2105  2885 25124 10520  2202  2961
 27828  6767  2228  1506 13550  2635  5556  4787  3781  3137  3818  3942
   914   401  2829  2977  4934  2062  2354  5455 15020  1424  3273 22040
  4416  3908 10566   991  4931  1086  7430  6497   114  7896  2346  3418
  3432  2907  1151  2414  2290 15831 41310  4508  2538  3456  6418  1848
  3887  5721  9562  1455  2036  1831 11678  2936  2993  7443  6360  1797
  1173  4687  6723  2009  6097  2653  1639 18481  7978  2387  5060]
=============================================================================================================
===========================================capital_loss=======================================================
[   0 2042 1408 1902 1573 1887 1719 1762 1564 2179 1816 1980 1977 1876
 1340 2206 1741 1485 2339 2415 1380 1721 2051 2377 1669 2352 1672  653
 2392 1504 2001 1590 1651 1628 1848 1740 2002 1579 2258 1602  419 2547
 2174 2205 1726 2444 1138 2238  625  213 1539  880 1668 1092 1594 3004
```

```
# Seperate both the dataset columns into numeric and categorical columns
numeric_cols = [feature for feature in income.columns if income[feature].dtype != 'O']
```

```
categorical_cols =[feature for feature in income.columns if income[feature].dtype == 'O']
```

```
print('Categorical Features')
print(categorical_cols)
print('=====================================================================================================================')
print('Numeric Features')
print(numeric_cols)
```

```
Categorical Features
['workclass', 'education', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'native_country', 'income_class']
=====================================================================================================================
Numeric Features
['age', 'fnlwgt', 'education-num', 'capital_gain', 'capital_loss', 'hour_per_week']
```

```
# Removing trailing spaces from all the categorical columns
for col in categorical_cols:
    income[col]=income[col].str.strip()
    print(income[col].unique())
```

```
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov' '?'
 'Self-emp-inc' 'Without-pay' 'Never-worked']
['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']
['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-absent'
 'Separated' 'Married-AF-spouse' 'Widowed']
['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']
['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-relative']
['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']
['Male' 'Female']
['United-States' 'Cuba' 'Jamaica' 'India' '?' 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']
['<=50K' '>50K']
```

```
# Cleaning up the some categorical columns by replacing Special xter with Most frequent
income['workclass'] = income['workclass'].astype(str)
income['occupation'] = income['occupation'].astype(str)
income['native_country'] = income['native_country'].astype(str)
income['education']=income['education'].astype(str)
income['workclass'] = income['workclass'].str.replace('?', "Private")
income['occupation'] = income['occupation'].str.replace('?', "Prof-specialty")
income['native_country'] = income['native_country'].str.replace('?', "United-States")
income['education'].replace({'11th':"HS-grad", '9th':"HS-grad", '7th-8th':"HS-grad", '1st-4th':"HS-grad",'5th-6th':"HS-grad",'10th':"H
```

```
income['workclass'].unique()
```

```
array(['State-gov', 'Self-emp-not-inc', 'Private', 'Federal-gov',
       'Local-gov', 'Self-emp-inc', 'Without-pay', 'Never-worked'],
      dtype=object)
```

```
income['occupation'].unique()
```

```
array(['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'], dtype=object)
```

```
income['education'].unique()
```

```
array(['Bachelors', 'HS-grad', 'Masters', 'Some-college', 'Assoc-acdm',
       'Assoc-voc', 'Doctorate', 'Prof-school', 'Preschool'], dtype=object)
```

```
income['native_country'].unique()
```

```
array(['United-States', 'Cuba', 'Jamaica', 'India', 'Mexico', 'South',
       'Puerto-Rico', 'Honduras', 'England', 'Canada', 'Germany', 'Iran',
       'Philippines', 'Italy', 'Poland', 'Columbia', 'Cambodia',
       'Thailand', 'Ecuador', 'Laos', 'Taiwan', 'Haiti', 'Portugal',
       'Dominican-Republic', 'El-Salvador', 'France', 'Guatemala',
       'China', 'Japan', 'Yugoslavia', 'Peru',
       'Outlying-US(Guam-USVI-etc)', 'Scotland', 'Trinadad&Tobago',
       'Greece', 'Nicaragua', 'Vietnam', 'Hong', 'Ireland', 'Hungary',
       'Holand-Netherlands'], dtype=object)
```

```
# Statistical analysis about the dataset
income.describe()
```

	age	fnlwgt	education-num	capital_gain	capital_loss	hour_per_week
count	32537.000000	3.253700e+04	32537.000000	32537.000000	32537.000000	32537.000000
mean	38.585549	1.897808e+05	10.081815	1078.443741	87.368227	40.440329
std	13.637984	1.055565e+05	2.571633	7387.957424	403.101833	12.346889
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.369930e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

```
income.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
age	32537.0	NaN	NaN	NaN	38.585549	13.637984	17.0	28.0	37.0	48.0	90.0
workclass	32537	8	Private	24509	NaN	NaN	NaN	NaN	NaN	NaN	NaN
fnlwgt	32537.0	NaN	NaN	NaN	189780.848511	105556.471009	12285.0	117827.0	178356.0	236993.0	1484705.0
education	32537	9	HS-grad	14692	NaN	NaN	NaN	NaN	NaN	NaN	NaN
education-num	32537.0	NaN	NaN	NaN	10.081815	2.571633	1.0	9.0	10.0	12.0	16.0
marital-status	32537	7	Married-civ-spouse	14970	NaN	NaN	NaN	NaN	NaN	NaN	NaN
occupation	32537	14	Prof-specialty	5979	NaN	NaN	NaN	NaN	NaN	NaN	NaN
relationship	32537	6	Husband	13187	NaN	NaN	NaN	NaN	NaN	NaN	NaN
race	32537	5	White	27795	NaN	NaN	NaN	NaN	NaN	NaN	NaN
sex	32537	2	Male	21775	NaN	NaN	NaN	NaN	NaN	NaN	NaN
capital_gain	32537.0	NaN	NaN	NaN	1078.443741	7387.957424	0.0	0.0	0.0	0.0	99999.0
capital_loss	32537.0	NaN	NaN	NaN	87.368227	403.101833	0.0	0.0	0.0	0.0	4356.0
hour_per_week	32537.0	NaN	NaN	NaN	40.440329	12.346889	1.0	40.0	40.0	45.0	99.0
native_country	32537	41	United-States	29735	NaN	NaN	NaN	NaN	NaN	NaN	NaN
income_class	32537	2	<=50K	24698	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
# Checking the correlation among the numeric columns
income[numeric_cols]
```

	age	fnlwgt	education-num	capital_gain	capital_loss	hour_per_week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40
...	...	...	...	...	...	...
32556	27	257302	12	0	0	38
32557	40	154374	9	0	0	40
32558	58	151910	9	0	0	40
32559	22	201490	9	0	0	20
32560	52	287927	9	15024	0	40

32537 rows × 6 columns

```
# Correlation among the numeric columns
income[numeric_cols].corr()
```

	age	fnlwgt	education-num	capital_gain	capital_loss	hour_per_week
age	1.000000	-0.076447	0.036224	0.077676	0.057745	0.068515
fnlwgt	-0.076447	1.000000	-0.043388	0.000429	-0.010260	-0.018898
education-num	0.036224	-0.043388	1.000000	0.122664	0.079892	0.148422
capital_gain	0.077676	0.000429	0.122664	1.000000	-0.031639	0.078408
capital_loss	0.057745	0.010260	0.079892	0.031639	1.000000	0.054339

```
# Checking missing values again after columns cleaning
income.isnull().sum()
#income['workclass'].isnull().sum()
#income[income['workclass'] == str(np.nan)]

```

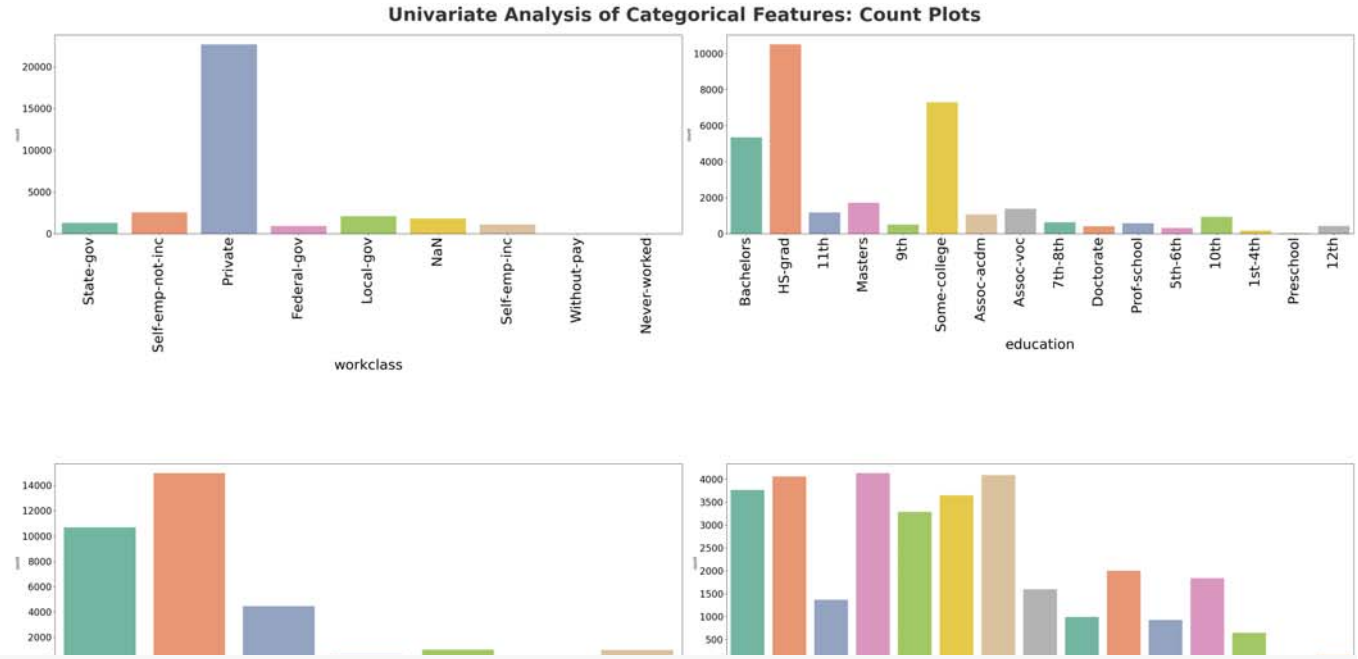
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hour_per_week     0
native_country    0
income_class      0
dtype: int64
```

```
for col in categorical_cols:
    print(f"{col}")
    print(f"{income[col].value_counts(normalize=True)*100}")
    print("================================================")
```

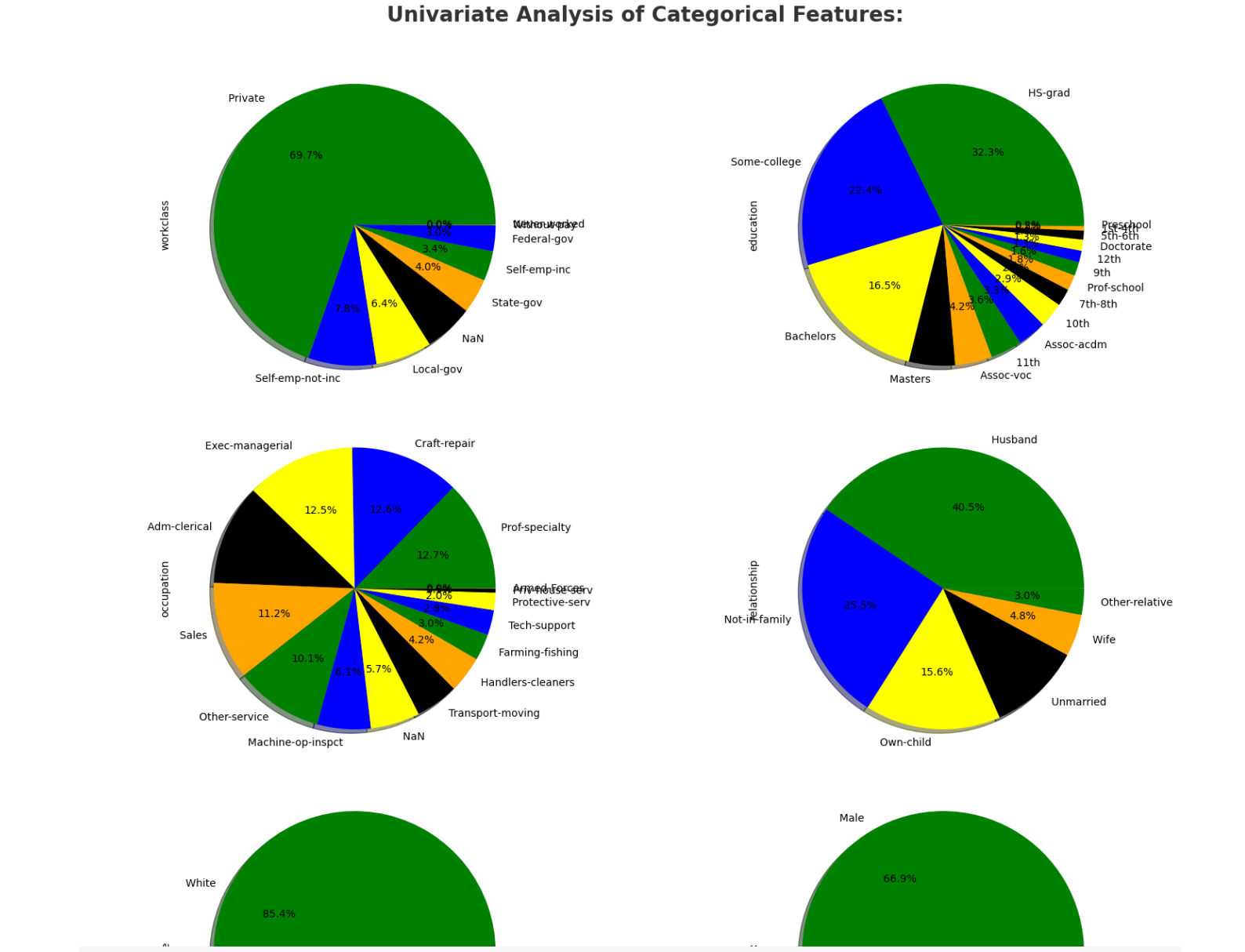
```
workclass
Private             75.326551
Self-emp-not-inc     7.806497
Local-gov            6.432677
State-gov            3.989304
Self-emp-inc         3.429941
Federal-gov          2.950487
Without-pay          0.043028
Never-worked         0.021514
Name: workclass, dtype: float64
================================================
education
HS-grad         45.154747
Some-college    22.380674
Bachelors       16.452039
Masters          5.292436
Assoc-voc        4.247472
Assoc-acdm       3.279344
Prof-school      1.770292
Doctorate        1.269324
Preschool        0.153671
Name: education, dtype: float64
================================================
marital_status
Married-civ-spouse       46.009159
Never-married            32.784215
Divorced                 13.649076
Separated                 3.150260
Widowed                   3.051910
Married-spouse-absent     1.284691
Married-AF-spouse         0.070689
Name: marital_status, dtype: float64
================================================
occupation
Prof-specialty       18.376003
Craft-repair         12.582598
Exec-managerial      12.493469
Adm-clerical         11.580662
Sales                11.217998
Other-service        10.114639
Machine-op-inspct     6.146848
Transport-moving      4.908258
Handlers-cleaners     4.207518
Farming-fishing       3.048837
Tech-support          2.849064
Protective-serv       1.994652
Priv-house-serv       0.451793
Armed-Forces          0.027661
Name: occupation, dtype: float64
```

```
==========================================
relationship
Husband           40.529244
Not-in-family     25.484833
Own-child         15.563820
Unmarried         10.587946
Wife               4.819129
Other-relative     3.015029
Name: relationship, dtype: float64
```

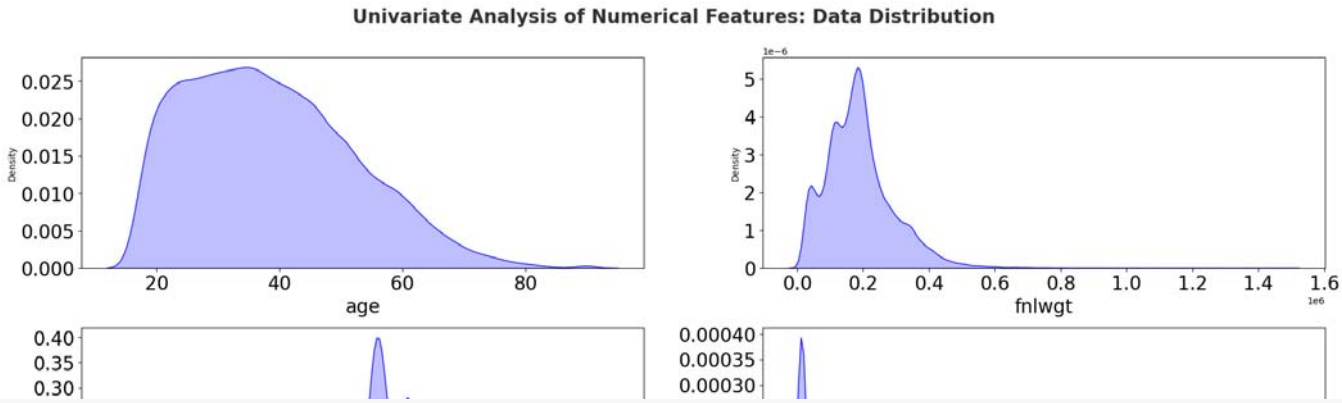
```
plt.figure(figsize=(40, 60))
plt.suptitle('Univariate Analysis of Categorical Features: Count Plots', fontsize=40, fontweight='bold', alpha=0.8, y=1.)
for i in range(0, len(categorical_cols)):
    plt.subplot(5, 2, i+1)
    sns.countplot(x=income[categorical_cols[i]],palette="Set2")
    plt.xlabel(categorical_cols[i], fontsize = 30)
    plt.xticks(rotation=90, fontsize = 30)
    plt.yticks(fontsize = 20)
    plt.tight_layout()

```



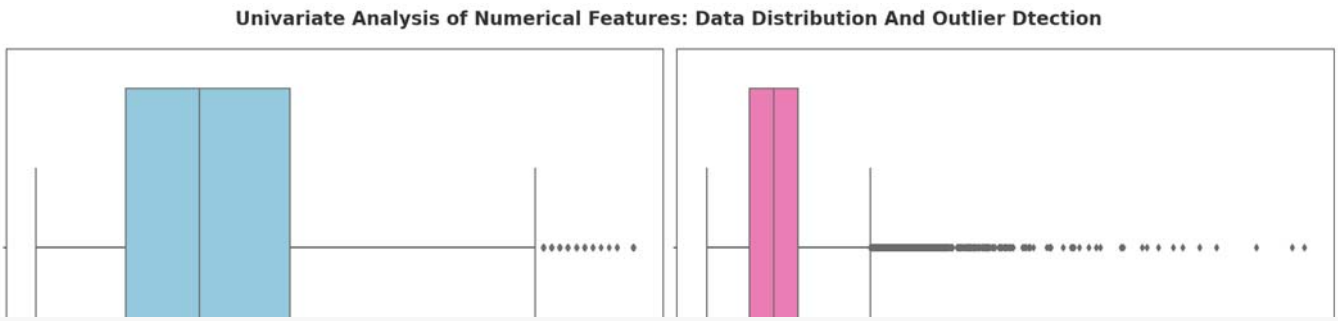

```
plt.figure(figsize=(15, 20))
plt.suptitle('Univariate Analysis of Categorical Features: ', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
plt.subplot(421)
income['workclass'].value_counts().plot.pie(y=income['workclass'],autopct='%1.1f%%',shadow=True,colors=['green','blue','yellow','black
plt.subplot(422)
income['education'].value_counts().plot.pie(y=income['education'],autopct='%1.1f%%',shadow=True,colors=['green','blue','yellow','black
plt.subplot(423)
income['occupation'].value_counts().plot.pie(y=income['occupation'],autopct='%1.1f%%',shadow=True,colors=['green','blue','yellow','bla
plt.subplot(424)
income['relationship'].value_counts().plot.pie(y=income['relationship'],autopct='%1.1f%%',shadow=True,colors=['green','blue','yellow',
plt.subplot(425)
income['race'].value_counts().plot.pie(y=income['race'],autopct='%1.1f%%',shadow=True,colors=['green','blue','yellow','black', 'orange
plt.subplot(426)
income['sex'].value_counts().plot.pie(y=income['sex'],autopct='%1.1f%%',shadow=True,colors=['green','blue','yellow','black', 'orange']
plt.subplot(427)
income['native_country'].value_counts().plot.pie(y=income['native_country'],autopct='%1.1f%%',shadow=True,colors=['green','blue','yell
plt.tight_layout()
plt.show()

```



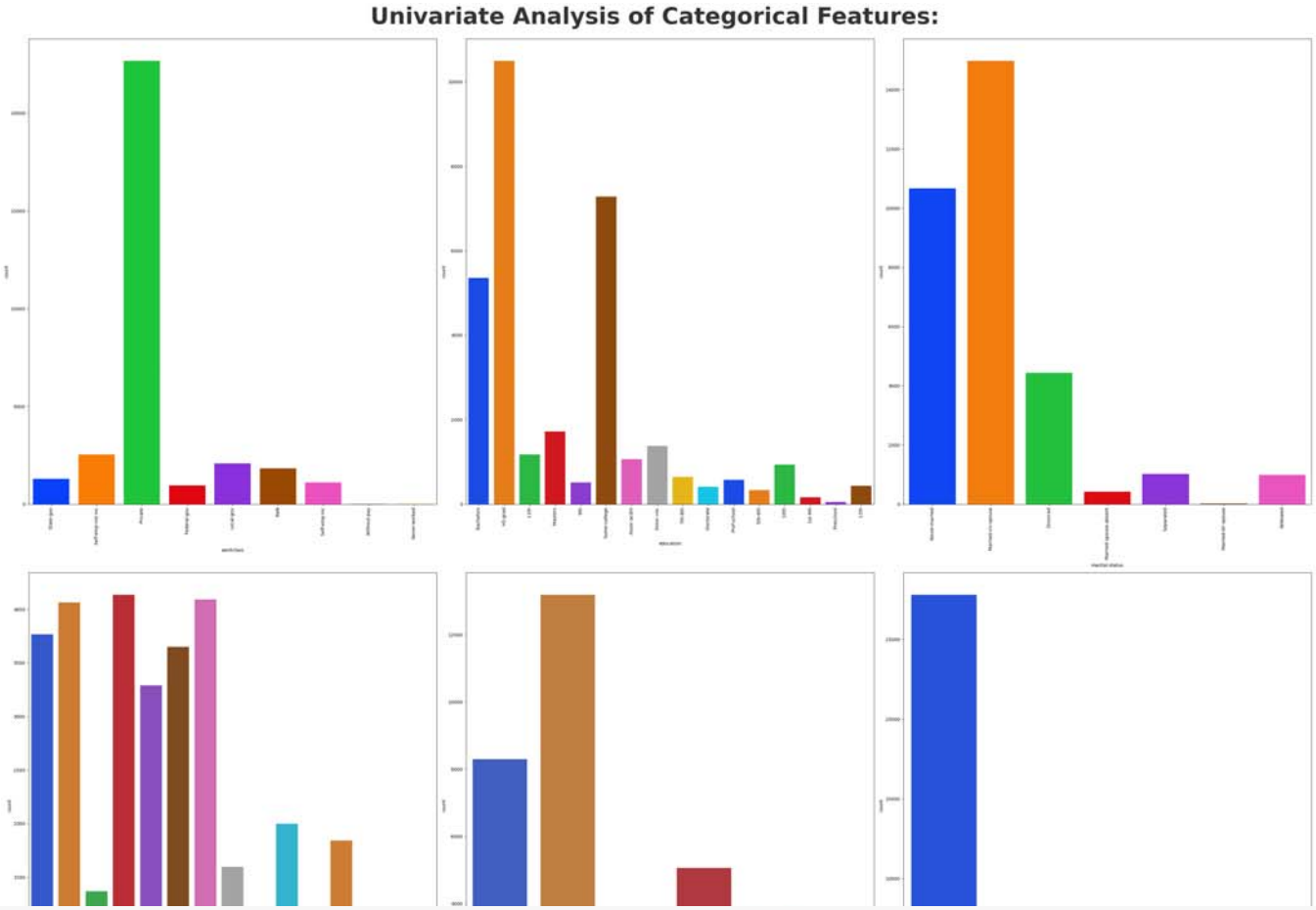

```
plt.figure(figsize=(20, 20))
plt.suptitle('Univariate Analysis of Numerical Features: Data Distribution', fontsize=20, fontweight='bold', alpha=0.8, y=1.)

for i in range(0, len(numeric_cols)):
    plt.subplot(5, 2, i+1)
    sns.kdeplot(x=income[numeric_cols[i]],shade=True, color='b')
    plt.xlabel(numeric_cols[i], fontsize = 20)
    plt.xticks(fontsize = 20)
    plt.yticks(fontsize = 20)
    plt.tight_layout()
```




```
1
2 plt.subplots(3,2,figsize=(20,20))
3 plt.suptitle('Univariate Analysis of Numerical Features: Data Distribution And Outlier Dtection', fontsize=20, fontweight='bold', alph
4 plt.subplot(321)
5 sns.boxplot(income['age'],color='skyblue')
6 plt.subplot(322)
7 sns.boxplot(income['fnlwgt'],color='hotpink')
8 plt.subplot(323)
9 sns.boxplot(income['education-num'],color='yellow')
10 plt.subplot(324)
11 sns.boxplot(income['capital_gain'],color='lightgreen')
12 plt.subplot(325)
13 sns.boxplot(income['capital_loss'],color='lightblue')
14 plt.subplot(326)
15 sns.boxplot(income['hour_per_week'],color='orange')
16 plt.tight_layout()
17 plt.show()
18
```

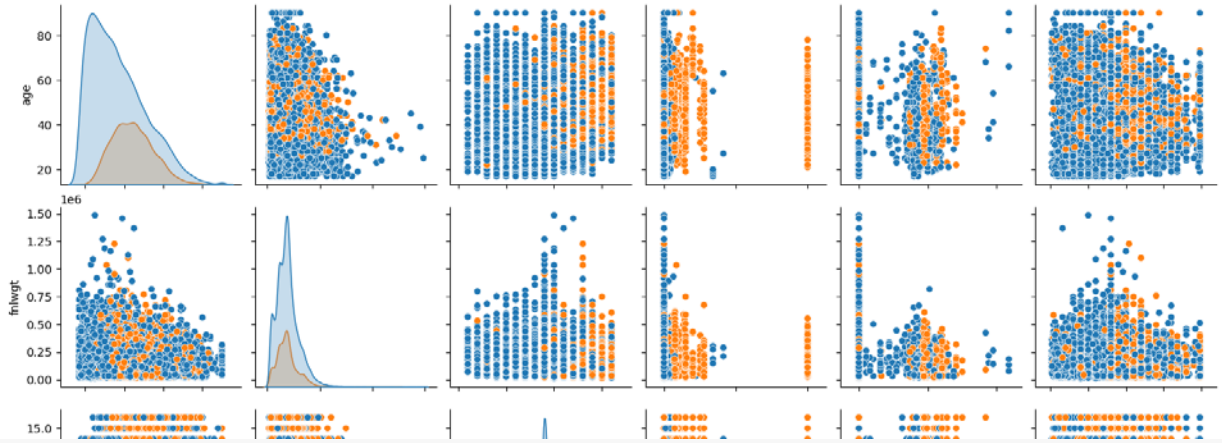



```
#
plt.subplots(3,3,figsize=(40,50))
plt.suptitle('Univariate Analysis of Categorical Features: ', fontsize=50, fontweight='bold', alpha=0.8, y=1.)
plt.xticks(rotation=90)
plt.subplot(331)
sns.countplot(x=income['workclass'],data=income,palette = 'bright',saturation=0.95)
plt.xticks(rotation=90)
plt.subplot(332)
sns.countplot(x=income['education'],data=income,palette = 'bright',saturation=0.80)
plt.xticks(rotation=90)
plt.subplot(333)
sns.countplot(x=income['marital-status'],data=income,palette = 'bright',saturation=0.90)
plt.xticks(rotation=90)
plt.subplot(334)
sns.countplot(x=income['occupation'],data=income,palette = 'bright',saturation=0.60)
plt.xticks(rotation=90)
plt.subplot(335)
sns.countplot(x=income['relationship'],data=income,palette = 'bright',saturation=0.50)
plt.xticks(rotation=90)
plt.subplot(336)
sns.countplot(x=income['race'],data=income,palette = 'bright',saturation=0.70)
plt.xticks(rotation=90)
plt.subplot(337)
sns.countplot(x=income['sex'],data=income,palette = 'bright',saturation=0.85)
plt.xticks(rotation=90)
plt.subplot(338)
sns.countplot(x=income['native_country'],data=income,palette = 'bright',saturation=0.40)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

```




```
# Correlation among numeric features
plt.figure(figsize=(40, 40))
plt.title('MultiVariate Analysis: Feature Correlation', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
sns.pairplot(income, hue='income_class')
plt.tight_layout()
plt.show()

```

```
<seaborn.axisgrid.PairGrid at 0x2a110a01670>
<Figure size 4000x4000 with 0 Axes>
```




```
# Checkin the relationship between numeric columns and the label feature
plt.figure(figsize=(20,20))
plt.subplot(3,2,1)
sns.barplot (x=income['age'], y=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,2)
sns.barplot (x=income['fnlwgt'], y=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,3)
sns.barplot (x=income['education-num'], y=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,4)
sns.barplot (x=income['capital_gain'], y=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,5)
sns.barplot (x=income['capital_loss'], y=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,6)
sns.barplot (x=income['hour_per_week'], y=income['income_class'], hue=income['income_class'])


```

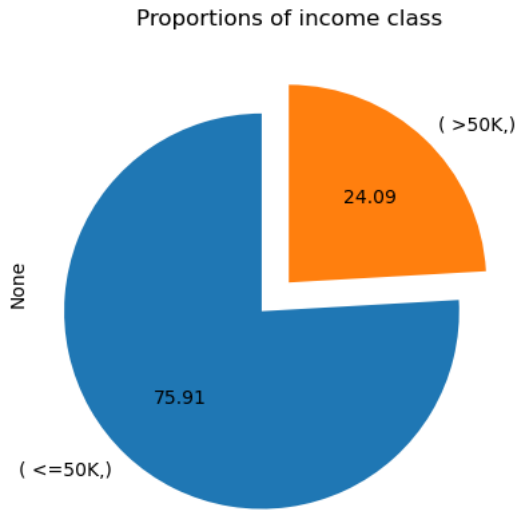
```
<AxesSubplot:xlabel='hour_per_week', ylabel='income_class'>
```


```
# Checkin the relationship between numeric columns and the label feature
plt.figure(figsize=(20,20))
plt.subplot(3,2,1)
sns.boxplot (y=income['age'], x=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,2)
sns.boxplot (y=income['fnlwgt'], x=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,3)
sns.boxplot (y=income['education-num'], x=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,4)
sns.boxplot (y=income['capital_gain'], x=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,5)
sns.boxplot (y=income['capital_loss'], x=income['income_class'], hue=income['income_class'])
plt.subplot(3,2,6)
sns.boxplot (y=income['hour_per_week'], x=income['income_class'], hue=income['income_class'])
```

```
<AxesSubplot:xlabel='income_class', ylabel='hour_per_week'>
```


```
# Proportion of people that earn more than 50k
income.value_counts(['income_class']).plot.pie(y='income_class',startangle=90, explode=(0.2,0), title='Proportions of income class', l
```

```
<AxesSubplot:title={'center':'Proportions of income class'}, ylabel='None'>
```

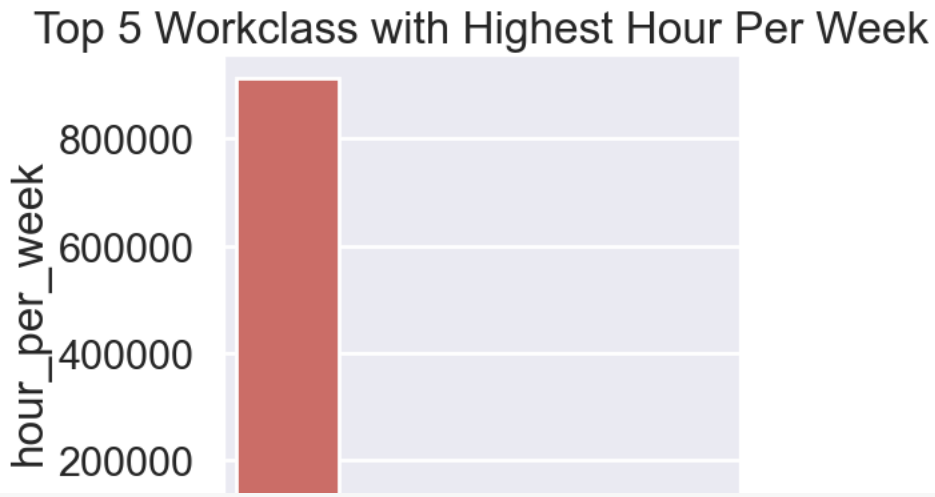



```
# Top 5 workclass earning highest hour per week
income.groupby(['workclass'])['hour_per_week'].sum().sort_values(ascending = False).reset_index()

```

	workclass	hour_per_week
0	Private	913065
1	Self-emp-not-inc	112836
2	Local-gov	85777
3	NaN	58604
4	Self-emp-inc	54481
5	State-gov	50663
6	Federal-gov	39724
7	Without-pay	458
8	Never-worked	199

```
plt.figure(figsize=(5,5))
Top_5 =income.groupby(['workclass'])['hour_per_week'].sum().sort_values(ascending = False).reset_index()[:4]
#sns.set_context("poster")
sns.set_style("darkgrid")
plt.title('Top 5 Workclass with Highest Hour Per Week')
sns.barplot(data = Top_5, y='hour_per_week', x='workclass', palette='hls')
plt.xticks(rotation=90)
```

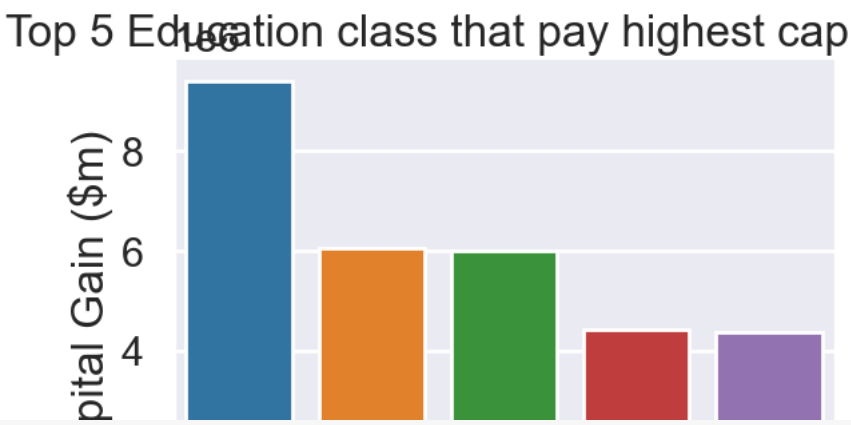
```
(array([0, 1, 2, 3]),
 [Text(0, 0, ' Private'),
  Text(1, 0, ' Self-emp-not-inc'),
  Text(2, 0, ' Local-gov'),
  Text(3, 0, ' NaN')])
```



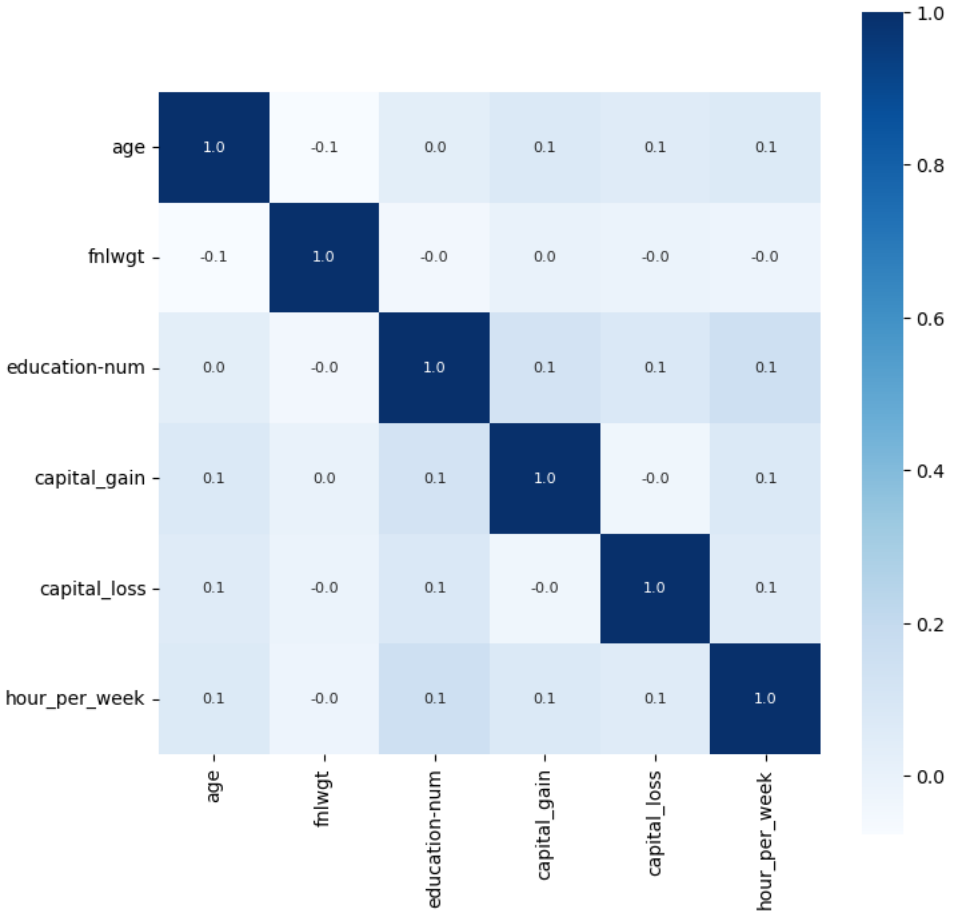

```
# Top 5 Education class that pay highest capital gain
income.groupby(['education'])['capital_gain'].sum().sort_values(ascending=False).reset_index()
```

	education	capital_gain
0	Bachelors	9404984
1	HS-grad	6056978
2	Prof-school	5998704
3	Masters	4415297
4	Some-college	4366027
5	Doctorate	1970070
6	Assoc-voc	988201
7	Assoc-acdm	683306
8	10th	377468
9	11th	252740
10	9th	175834
11	7th-8th	151125
12	12th	123010
13	5th-6th	58615
14	Preschool	45818
15	1st-4th	21147

```
Top_5_Cgain=income.groupby(['education'])['capital_gain'].sum().sort_values(ascending=False).reset_index()[:5]
plt.title('Top 5 Education class that pay highest capital gain')
sns.barplot(data=Top_5_Cgain, x = 'education', y='capital_gain')
plt.xticks(rotation=45)
plt.ylabel('Capital Gain ($m)')
plt.show()
```




```
plt.figure(figsize=(8,8))
sns.heatmap(income.corr(),cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Blues')
plt.show()
```




```
sns.pairplot(income, hue='income_class')
plt.show()
```

- Feature Engineering involves handling the following
  - missing values
  - handling outlier
  - handling categorical encoding
  - feature scaling
  - feature selection

```
# Check the categorical columns propability of occurence/frequency/count
for col in categorical_cols:
    print(f"{col}")
    print(f"{income[col].value_counts(normalize=True)*100}")
    print("================================================")
```

```
Portugal                    0.113717
Nicaragua                   0.104496
Peru                        0.095276
France                      0.089129
Greece                      0.089129
Ecuador                     0.086056
Ireland                     0.073762
Hong                        0.061468
Cambodia                    0.058395
Trinadad&Tobago             0.058395
Laos                        0.055322
Thailand                    0.055322
Yugoslavia                  0.049175
Outlying-US(Guam-USVI-etc)  0.043028
Honduras                    0.039955
Hungary                     0.039955
Scotland                    0.036881
Holand-Netherlands          0.003073
Name: native_country, dtype: float64
==========================================
income_class
<=50K    75.907428
>50K     24.092572
Name: income_class, dtype: float64
==========================================
```

```
# Handling Columns Encoding
ordinal_cols =["education","relationship","marital_status","workclass"]
oho_cols = ["native_country","sex","race","occupation"]

# Ranking of ordinal_cols
education = ['Preschool','HS-grad','Some-college','Assoc-voc','Assoc-acdm','Bachelors','Masters','Prof-school',
 'Doctorate'  ]
relationship = ['Not-in-family','Other-relative','Unmarried','Own-child','Wife','Husband']
marital_status = ['Never-married', 'Married-civ-spouse', 'Divorced' ,'Married-spouse-absent'
 'Separated', 'Married-AF-spouse', 'Widowed']
workclass =['Never-worked','Without-pay','Self-emp-inc','Local-gov','Federal-gov','Private','Self-emp-not-inc','State-gov']

```

```
# Ordinal Encoding Using Replace Function
income['education'].replace({'Preschool':0,'HS-grad':1,'Some-college':2,'Assoc-voc':3,'Assoc-acdm':4,'Bachelors':5,'Masters':6,'Prof-s
 'Doctorate':8 }, inplace=True)
# For relationship column
income['relationship'].replace({'Not-in-family':0,'Other-relative':1,'Unmarried':2,'Own-child':3,'Wife':4,'Husband':5 }, inplace=True)
# For marital_status
income['marital_status'].replace({'Never-married':0, 'Married-civ-spouse':1, 'Divorced':2 ,'Married-spouse-absent':3,
 'Separated':4, 'Married-AF-spouse':5, 'Widowed':6 }, inplace=True)
# For workclass
income['workclass'].replace({'Never-worked':0,'Without-pay':1,'Self-emp-inc':2,'Local-gov':3,'Federal-gov':4,'Private':5,'Self-emp-not
```

```
# Check The Encoding
for ord in ordinal_cols:
    print(f"The ordinal encoding result for {ord} is: {income[ord].unique()}")
```

```
The ordinal encoding result for education is: [5 1 6 2 4 3 8 7 0]
The ordinal encoding result for relationship is: [0 5 4 3 2 1]
The ordinal encoding result for marital_status is: [0 1 2 3 4 5 6]
The ordinal encoding result for workclass is: [7 6 5 4 3 2 1 0]
```

```
# OneHot Encoding Using Python Function

# For Sex column
income['sex'] = np.where(income["sex"]=="Female",0,1)
print(income['sex'].unique())

# For Native_Country column
# American has almost 92% of the entire dataset. To aviod unncessary multiplicity of columns using
# onehot encoding, i have basically encoded america with "1" while other nation with less than 8% with "0"
income['native_country'] = np.where(income["native_country"]=="United-States",1,0)
print(income['native_country'].unique())

```

```
[1 0]
[1 0]
```

```

income.head()
```

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	h
0	39	7	77516	5	13	0	Adm-clerical	0	White	1	2174	0	
1	50	6	83311	5	13	1	Exec-managerial	5	White	1	0	0	
2	38	5	215646	1	9	2	Handlers-cleaners	0	White	1	0	0	
3	53	5	234721	1	7	1	Handlers-cleaners	5	Black	1	0	0	

```
# OneHotEncoding for other columns

# For Race and Occupation
ohe=pd.get_dummies(income[["race","occupation"]],drop_first=True)
ohe
```

	race_Asian-Pac-Islander	race_Black	race_Other	race_White	occupation_Armed-Forces	occupation_Craft-repair	occupation_Exec-managerial	occupation_Farming-fishing
0	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	1	0
2	0	0	0	1	0	0	0	0
3	0	1	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
32556	0	0	0	1	0	0	0	0
32557	0	0	0	1	0	0	0	0
32558	0	0	0	1	0	0	0	0
32559	0	0	0	1	0	0	0	0
32560	0	0	0	1	0	0	1	0

32537 rows × 17 columns

```
# Concatenate original and the dummy dataset
income= pd.concat([income,ohe],axis="columns")
income.head()
```

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	...	occupation_Farming-fishing	occ
0	39	7	77516	5	13	0	Adm-clerical	0	White	1	...	0	
1	50	6	83311	5	13	1	Exec-managerial	5	White	1	...	0	
2	38	5	215646	1	9	2	Handlers-cleaners	0	White	1	...	0	
3	53	5	234721	1	7	1	Handlers-cleaners	5	Black	1	...	0	
4	28	5	338409	5	13	1	Prof-specialty	4	Black	0	...	0	

5 rows × 32 columns

```
income.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hour_per_week', 'native_country',
       'income_class', 'race_Asian-Pac-Islander', 'race_Black', 'race_Other',
       'race_White', 'occupation_Armed-Forces', 'occupation_Craft-repair',
       'occupation_Exec-managerial', 'occupation_Farming-fishing',
       'occupation_Handlers-cleaners', 'occupation_Machine-op-inspct',
       'occupation_Other-service', 'occupation_Priv-house-serv',
       'occupation_Prof-specialty', 'occupation_Protective-serv',
       'occupation_Sales', 'occupation_Tech-support',
```

```
       'occupation_Transport-moving'],
      dtype='object')
```

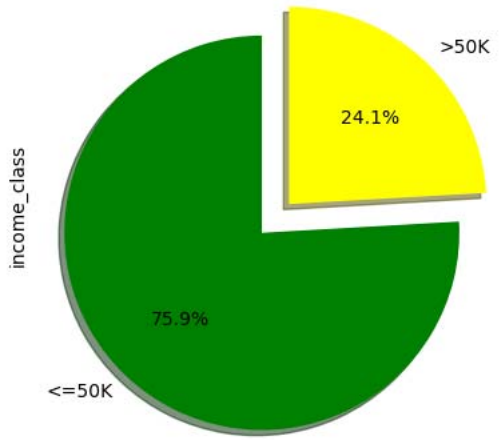
```
# Drop existing or original columns i.e Race and Occupation columns
income.drop(columns=["race","occupation"], axis = 1, inplace=True)
```

```
income.shape
```

```
(32537, 30)
```

```
# Calculating Lower and Upper Limit For columns And Capping Variables At Same
for col in df.columns[0:23]:
    lower_limit, upper_limit = find_limits(df,col)
    df[col]=np.where(df[col] > upper_limit, upper_limit,
                     np.where(df[col] < lower_limit, lower_limit, df[col]))
    print(f"For {col} variable: \n")
    print(f"The Lower Limit: {lower_limit}")
    print(f"The Upper Limit: {upper_limit}")
    print("==========================================================================================")

```

```
# Handling Imbalance Dataset
print(income["income_class"].value_counts())
income["income_class"].value_counts().plot.pie(y=income['income_class'],startangle=90, explode=(0.2,0),autopct='%1.1f%%',shadow=True,
plt.show()
```




```
# Convert the target column categories into "0" and "1"
income['income_class']=income['income_class'].apply(lambda x:1 if x == ">50K" else 0)
income['income_class'].unique()
```

```
array([0, 1], dtype=int64)
```

```
# From the above, the dataset is not balanced and we need to oversample the imbalance class using SMOTE library
from imblearn.over_sampling import SMOTE

```

```
X = income.drop('income_class',axis=1)
y = income['income_class']
```

```
X_res,y_res = SMOTE(random_state=42).fit_resample(X,y)
```

```
X_res
```

	age	workclass	fnlwgt	education	education-num	marital_status	relationship	sex	capital_gain	capital_loss	...	occupation
0	39	7	77516	5	13	0	0	1	2174	0	...	
1	50	6	83311	5	13	1	5	1	0	0	...	
2	38	5	215646	1	9	2	0	1	0	0	...	
3	53	5	234721	1	7	1	5	1	0	0	...	
4	28	5	338409	5	13	1	4	0	0	0	...	

```
y_res
```

```
0        0
1        0
2        0
3        0
4        0
        ..
49391    1
49392    1
49393    1
49394    1
49395    1
Name: income_class, Length: 49396, dtype: int64
```

```
# Let us combine the X_res and y_res back for further analysis after the oversampling
new_income=X_res
new_income["income_class"]=y_res
new_income
```

	age	workclass	fnlwgt	education	education-num	marital_status	relationship	sex	capital_gain	capital_loss	...	occupation
0	39	7	77516	5	13	0	0	1	2174	0	...	
1	50	6	83311	5	13	1	5	1	0	0	...	
2	38	5	215646	1	9	2	0	1	0	0	...	
3	53	5	234721	1	7	1	5	1	0	0	...	
4	28	5	338409	5	13	1	4	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
49391	43	5	280509	4	12	1	4	0	15024	0	...	
49392	43	4	198096	4	12	1	5	1	7688	0	...	
49393	48	4	242222	4	12	1	5	1	4386	0	...	
49394	43	5	332633	4	12	0	4	1	6868	0	...	
49395	60	5	238913	4	12	1	5	1	0	0	...	

49396 rows × 30 columns

```
# Checking the dataset balance after ovemsampling
# Handling Imbalance Dataset
print(new_income["income_class"].value_counts())
new_income["income_class"].value_counts().plot.pie(y=new_income['income_class'],startangle=90, explode=(0.2,0),autopct='%1.1f%%',shadc
plt.show()
```

```
0    24698
1    24698
Name: income_class, dtype: int64
```

```
# The education_num needs to be droped since we already we have a more encoded and
# representative of same in education column
new_income.drop("education-num",axis=1, inplace=True)
new_income.shape

```

```
(49396, 29)
```


```
# Check info about the new resampled dataset
new_income.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49396 entries, 0 to 49395
Data columns (total 29 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   age                           49396 non-null  int64
 1   workclass                     49396 non-null  int64
 2   fnlwgt                        49396 non-null  int64
 3   education                     49396 non-null  int64
 4   marital_status                49396 non-null  int64
 5   relationship                  49396 non-null  int64
 6   sex                           49396 non-null  int32
 7   capital_gain                  49396 non-null  int64
 8   capital_loss                  49396 non-null  int64
 9   hour_per_week                 49396 non-null  int64
 10  native_country                49396 non-null  int32
 11  race_Asian-Pac-Islander       49396 non-null  uint8
 12  race_Black                    49396 non-null  uint8
 13  race_Other                    49396 non-null  uint8
 14  race_White                    49396 non-null  uint8
 15  occupation_Armed-Forces       49396 non-null  uint8
 16  occupation_Craft-repair       49396 non-null  uint8
 17  occupation_Exec-managerial    49396 non-null  uint8
 18  occupation_Farming-fishing    49396 non-null  uint8
 19  occupation_Handlers-cleaners  49396 non-null  uint8
 20  occupation_Machine-op-inspct  49396 non-null  uint8
 21  occupation_Other-service      49396 non-null  uint8
 22  occupation_Priv-house-serv    49396 non-null  uint8
 23  occupation_Prof-specialty     49396 non-null  uint8
 24  occupation_Protective-serv    49396 non-null  uint8
 25  occupation_Sales              49396 non-null  uint8
 26  occupation_Tech-support       49396 non-null  uint8
 27  occupation_Transport-moving   49396 non-null  uint8
 28  income_class                  49396 non-null  int64
dtypes: int32(2), int64(10), uint8(17)
memory usage: 4.9 MB
```

```
# Save the cleaned dataset to a csv file named "final.csv"
new_income.to_csv("final.csv")
```

Model Building, Hyper-parameter Tuning And Model Evaluation

```
# Import The Key Packages
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# Spliting Dataset Into Dependent And Independent Features
X =new_income.drop("income_class", axis =1)
y = new_income["income_class"]
```

```
# Split Dataset Into Train And Test Splits
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2,random_state=42)
```

```
# Dataset Scaling And Standardisation
data_scaling = StandardScaler()
X_train =data_scaling.fit_transform(X_train)
X_test = data_scaling.transform(X_test)
```

```
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
```

```
print(f"X_train shape: {y_train.shape}")
print(f"X_test shape: {y_test.shape}")
```

```
X_train shape: (39516, 28)
X_test shape: (9880, 28)
X_train shape: (39516,)
X_test shape: (9880,)
```

Logistic Regression Without Hyper-parameter

```
# Model Building
log_regression=LogisticRegression(verbose=False)
log_regression.fit(X_train, y_train)
y_pred = log_regression.predict(X_test)
```

```
# Model Evaluation
print("=====================================================================================================")
print("Accuracy_Score")
print(accuracy_score(y_test, y_pred))
print("=====================================================================================================")
print(classification_report(y_test, y_pred))
```

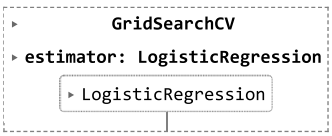
```
=====================================================================================================
Accuracy_Score
0.8545546558704453
=====================================================================================================
              precision    recall  f1-score   support

           0       0.85      0.86      0.86      4963
           1       0.86      0.85      0.85      4917

    accuracy                           0.85      9880
   macro avg       0.85      0.85      0.85      9880
weighted avg       0.85      0.85      0.85      9880
```

Logistic Regression With Hyper-parameter Tunning Using GridSearchCV

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

```
param_grid = {
    "penalty": ["l1", "l2","elasticnet"],
    "C" : [0.001, 0.01, 0.1, 1, 10],
    "solver" : ["lbfgs", "liblinear", "newton-cg", "sag", "saga"],
    "max_iter" : [100, 300, 500,1000,1500]
}
```

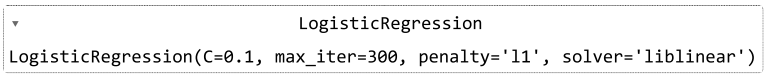
```
# Perform grid search with cross-validation
grid_search = GridSearchCV(log_regression, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
```




```
# Print the best hyperparameters
print("Best hyperparameters: ", grid_search.best_params_)

```

```
Best hyperparameters:  {'C': 0.1, 'max_iter': 300, 'penalty': 'l1', 'solver': 'liblinear'}
```

```
# Train the model with the best hyperparameters
best_model = LogisticRegression(**grid_search.best_params_)
best_model.fit(X_train, y_train)
```

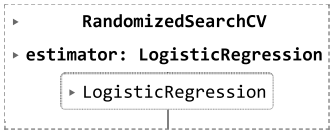



```
# Evaluate the performance on the validation(test) set
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy on validation set: {:.2f}".format(accuracy))
```

```
Accuracy on validation set: 0.85
```

Logistic Regression With Hyper-parameter Tunning Using RandomizedSearchCV

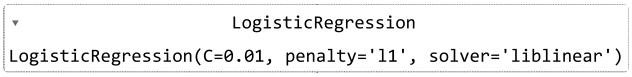
```
# Perform random search with cross-validation
random_search = RandomizedSearchCV(log_regression, param_grid, cv=5, scoring='accuracy', n_jobs=-1, n_iter=20, random_state=42, verbos
random_search.fit(X_train, y_train)

```




```
# Print the best hyperparameters
print("Best hyperparameters: ", random_search.best_params_)
```

```
Best hyperparameters:  {'solver': 'liblinear', 'penalty': 'l1', 'max_iter': 100, 'C': 0.01}
```

```
# Train the model with the best hyperparameters
best_model = LogisticRegression(**random_search.best_params_)
best_model.fit(X_train, y_train)
```




```
# Evaluate the performance on the validation set
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy on validation set: {:.2f}".format(accuracy))

```

```
Accuracy on validation set: 0.85
```

```

```