implementation of regression model

2148059

1)catherine is very curious about analysing the position of women in this modern era. she has an intuition that many bias exist when it comes to women's economic status. find out whether our assumption is correct or wrong.

in order to understand the existence of bias, take the help of adult dataset from uci repository, and come up with your conclusion.

the 2022 women's day theme is breaking the bias, through your research find out at what different the bias exist and what should be the solution to overcome it.

2)download any stock data, perform both single and multi linear regression

In [262]:
```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

In [263]:
```
data=pd.read_csv('/Users/persie/Downloads/adult.csv')
```

In [264]:
```
data.columns=['age',
'workclass',
'fnlwgt',
'education',
'education-num',
'marital-status',
'occupation',
'relationship',
'race',
'sex',
'capital-gain',
'capital-loss',
'hours-per-week',
'native-country','income' ]
```

In [265]:
```
data.head()
```

Out[265]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	inco
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=5
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=5
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=5
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=5
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=5

In [266]:
```
data.describe()
```

Out[266]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32560.000000	3.256000e+04	32560.000000	32560.000000	32560.000000	32560.000000
mean	38.581634	1.897818e+05	10.080590	1077.615172	87.306511	40.437469
std	13.640642	1.055498e+05	2.572709	7385.402999	402.966116	12.347618
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783630e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [267]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32560 entries, 0 to 32559
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32560 non-null  int64 
 1   workclass       32560 non-null  object
 2   fnlwgt          32560 non-null  int64 
 3   education       32560 non-null  object
 4   education-num   32560 non-null  int64 
 5   marital-status  32560 non-null  object
 6   occupation      32560 non-null  object
 7   relationship    32560 non-null  object
 8   race            32560 non-null  object
 9   sex             32560 non-null  object
 10  capital-gain    32560 non-null  int64 
 11  capital-loss    32560 non-null  int64 
 12  hours-per-week  32560 non-null  int64 
 13  native-country  32560 non-null  object
 14  income          32560 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [268]:
```
data.isnull().sum()
```
Out[268]:
```
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
hours-per-week    0
native-country    0
income            0
dtype: int64
```

there is no null values

splitting into numerical and categorical

In [269]:
```
numeric=data.select_dtypes(include=np.number).columns.tolist()
```
In [270]:
```
numeric
```
Out[270]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week']
```

In [271]:
```
category=data.select_dtypes(exclude=np.number).columns.tolist()
```

In [272]:
```
category
```

Out[272]:
```
['workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native-country',
 'income']
```

In [273]:
```
num=data[numeric]
cat=data[category]
```

In [274]:
```
num.head()
```

Out[274]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
0	50	83311	13	0	0	13
1	38	215646	9	0	0	40
2	53	234721	7	0	0	40
3	28	338409	13	0	0	40
4	37	284582	14	0	0	40

In [275]:
```
cat.head()
```

Out[275]:

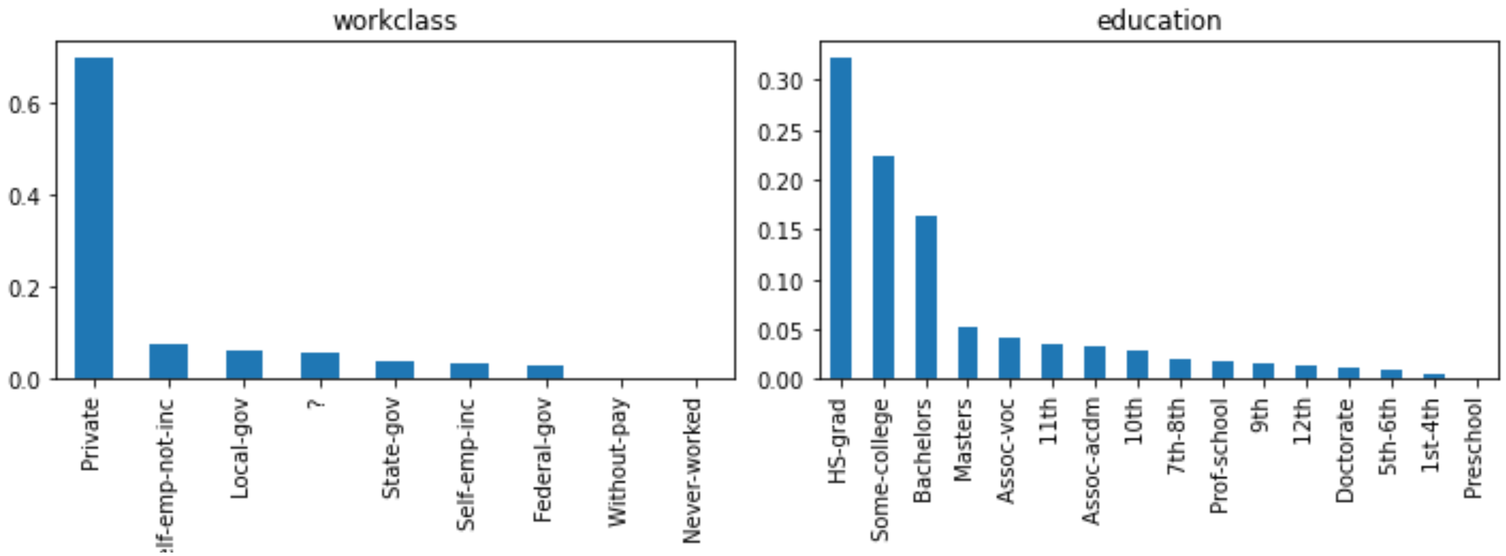
	workclass	education	marital-status	occupation	relationship	race	sex	native-country	income
0	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	<=50K
1	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States	<=50K
2	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States	<=50K
3	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba	<=50K
4	Private	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	United-States	<=50K

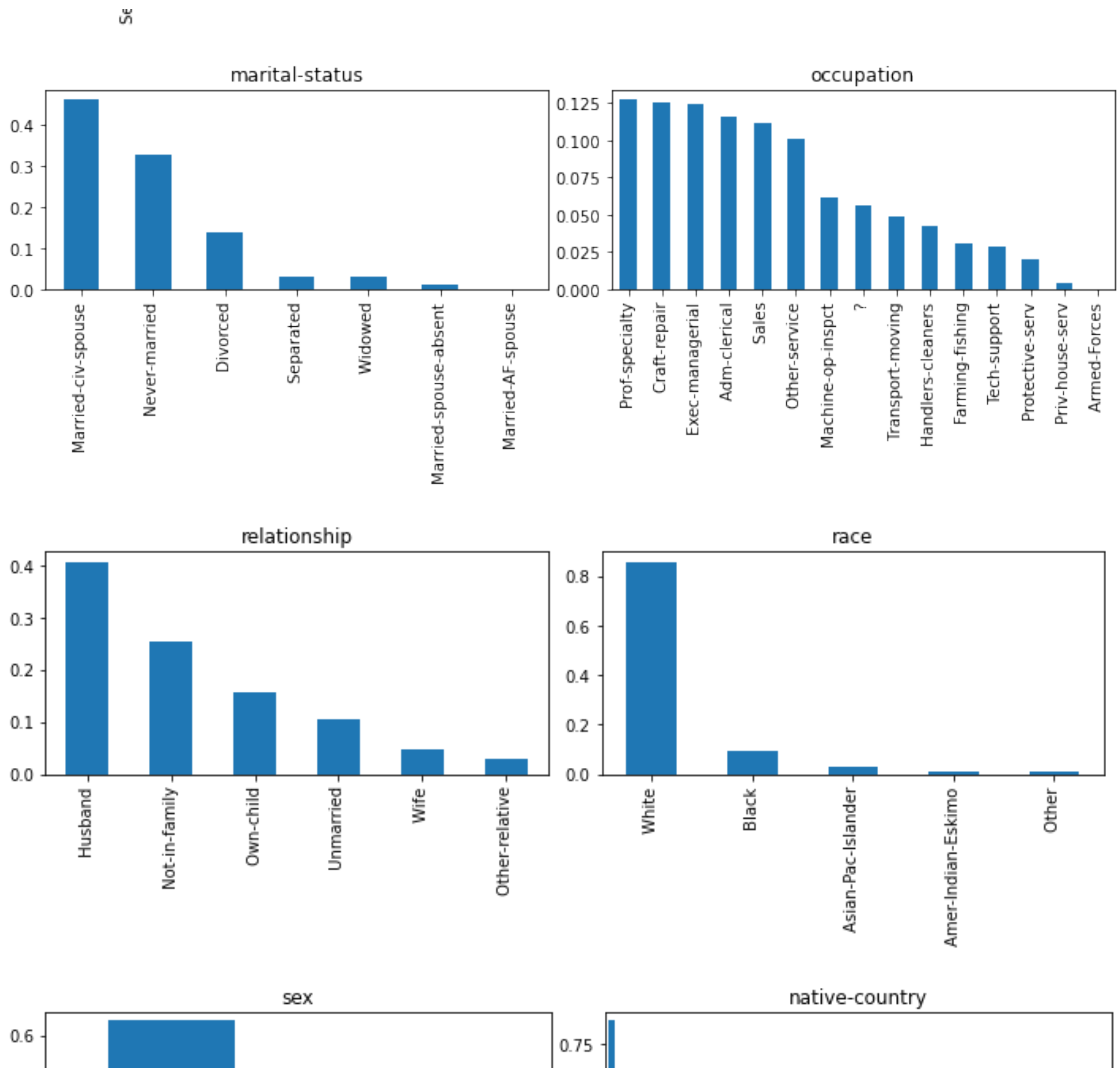
EDA

In [276]:
```
import matplotlib.pyplot as plt
```

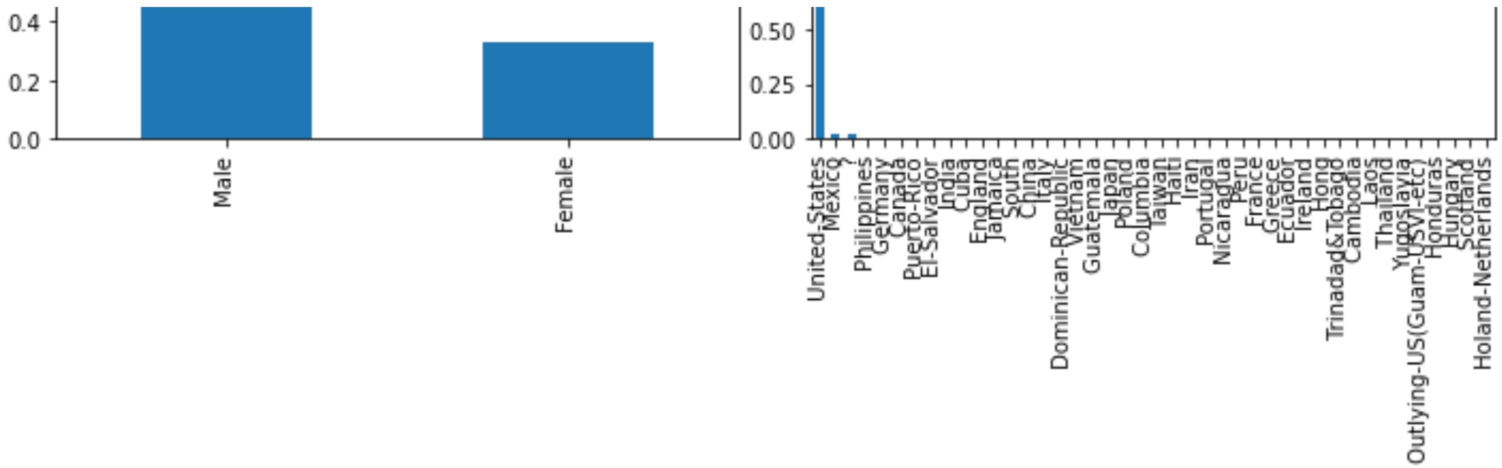
In [277]:
```python
# Function to perform univariate analysis of categorical columns
def plot_categorical_columns(dataframe):
    categorical_columns = dataframe.select_dtypes(include=['object']).columns

    for i in range(0,len(categorical_columns),2):
            if len(categorical_columns) > i+1:

                plt.figure(figsize=(10,4))
                plt.subplot(121)
                dataframe[categorical_columns[i]].value_counts(normalize=True).plot(kind='bar')
                plt.title(categorical_columns[i])
                plt.subplot(122)
                dataframe[categorical_columns[i+1]].value_counts(normalize=True).plot(kind='bar')
                plt.title(categorical_columns[i+1])
                plt.tight_layout()
                plt.show()

            else:
                dataframe[categorical_columns[i]].value_counts(normalize=True).plot(kind='bar')
                plt.title(categorical_columns[i])
```

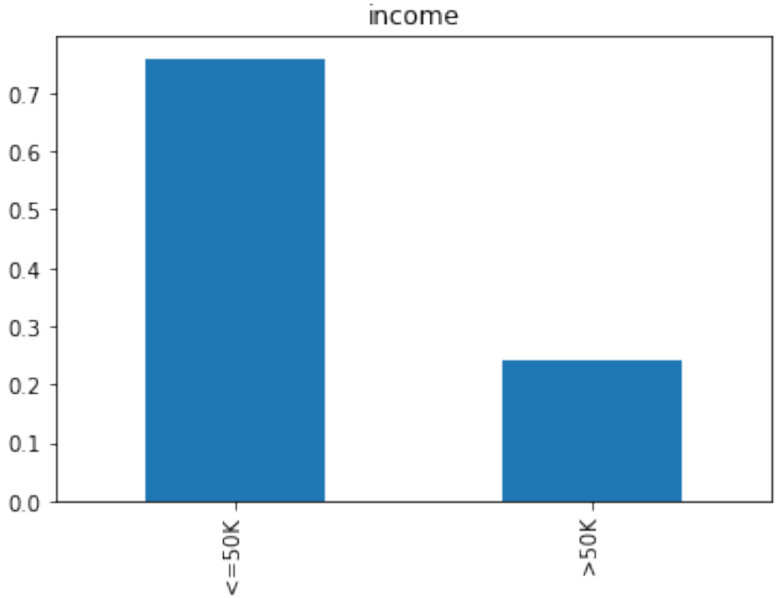
In [278]:
```python
plot_categorical_columns(cat)
```





marital-status
Married-civ-spouse
Never-married
Divorced
Separated
Widowed
Married-spouse-absent
Married-AF-spouse
occupation
Prof-specialty
Craft-repair
Exec-managerial
Adm-clerical
Sales
Other-service
Machine-op-inspct
?
Transport-moving
Handlers-cleaners
Farming-fishing
Tech-support
Protective-serv
Priv-house-serv
Armed-Forces
relationship
Husband
Not-in-family
Own-child
Unmarried
Wife
Other-relative
race
White
Black
Asian-Pac-Islander
Amer-Indian-Eskimo
Other
sex
native-country

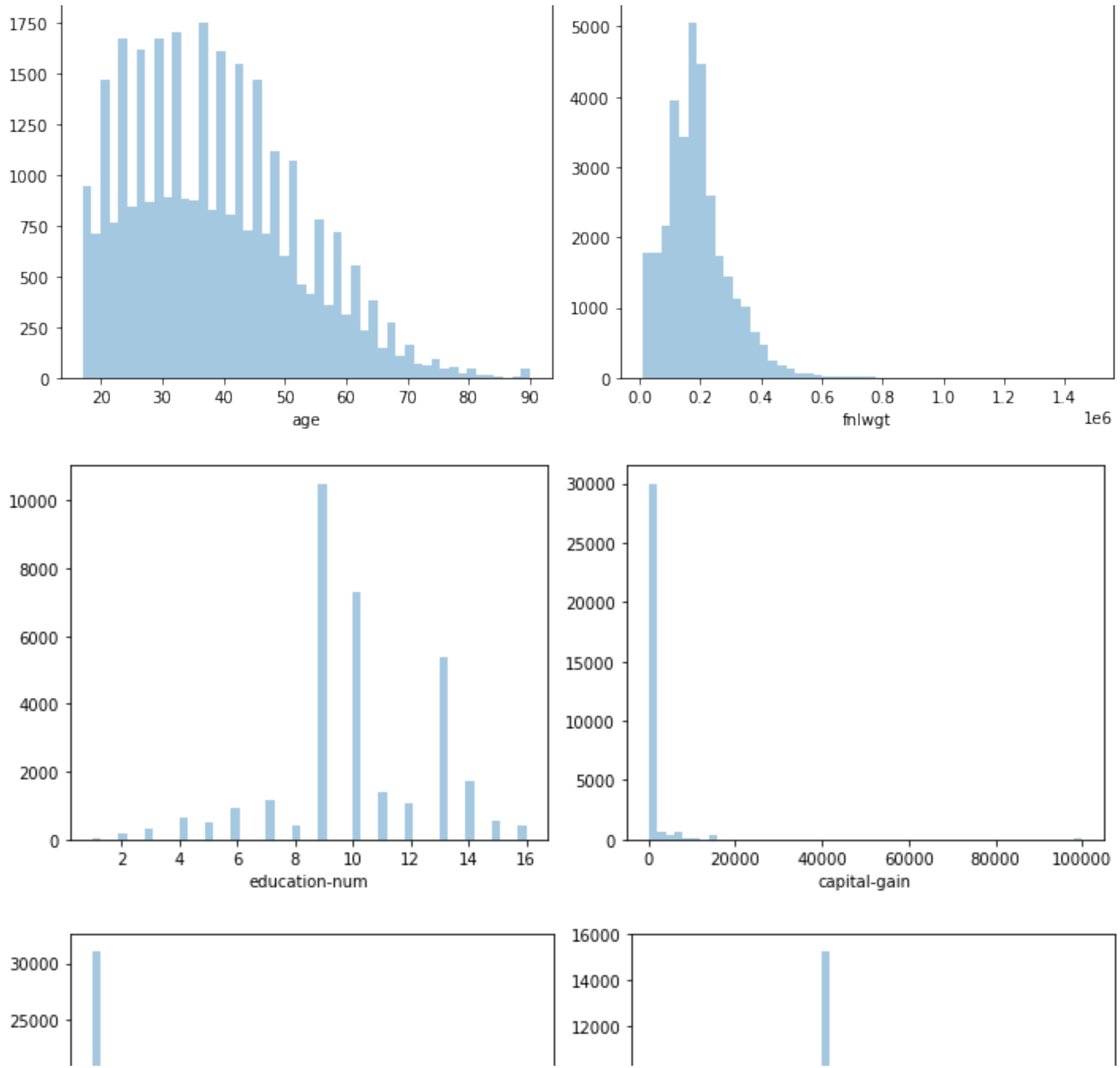


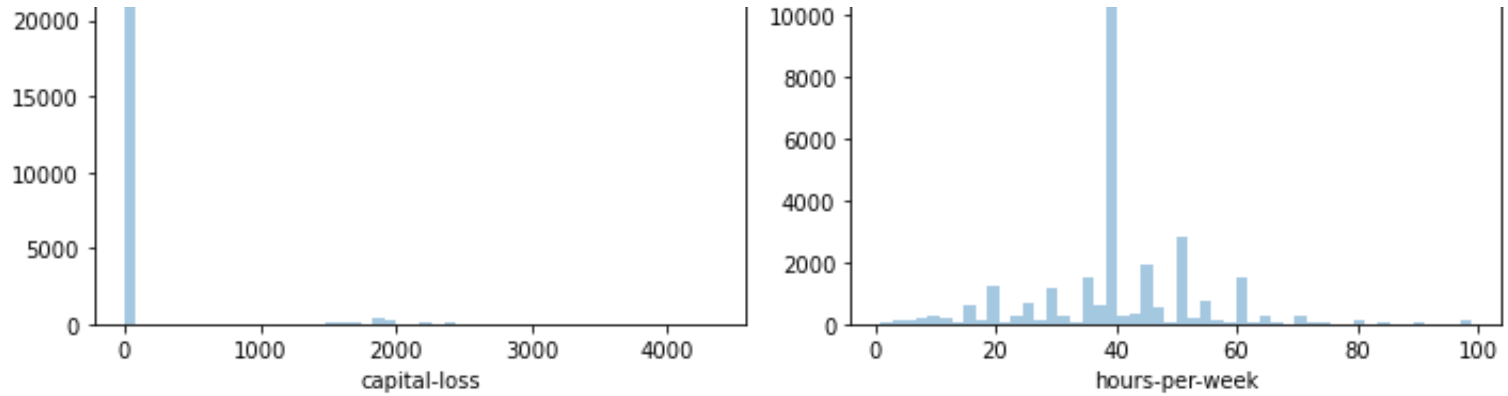


In [279]:
```
import seaborn as sns
```

In [280]:
```python
# Function to plot histograms
def plot_continuous_columns(dataframe):
    numeric_columns = dataframe.select_dtypes(include=['number']).columns.tolist()
    dataframe = dataframe[numeric_columns]

    for i in range(0,len(numeric_columns),2):
        if len(numeric_columns) > i+1:
            plt.figure(figsize=(10,4))
            plt.subplot(121)
            sns.distplot(dataframe[numeric_columns[i]], kde=False)
            plt.subplot(122)
            sns.distplot(dataframe[numeric_columns[i+1]], kde=False)
            plt.tight_layout()
            plt.show()

        else:
            sns.distplot(dataframe[numeric_columns[i]], kde=False)

# Function to plot boxplots
def plot_box_plots(dataframe):
    numeric_columns = dataframe.select_dtypes(include=['number']).columns.tolist()
    dataframe = dataframe[numeric_columns]

    for i in range(0,len(numeric_columns),2):
        if len(numeric_columns) > i+1:
            plt.figure(figsize=(10,4))
            plt.subplot(121)
            sns.boxplot(dataframe[numeric_columns[i]])
            plt.subplot(122)
            sns.boxplot(dataframe[numeric_columns[i+1]])
            plt.tight_layout()
            plt.show()

        else:
            sns.boxplot(dataframe[numeric_columns[i]])
```

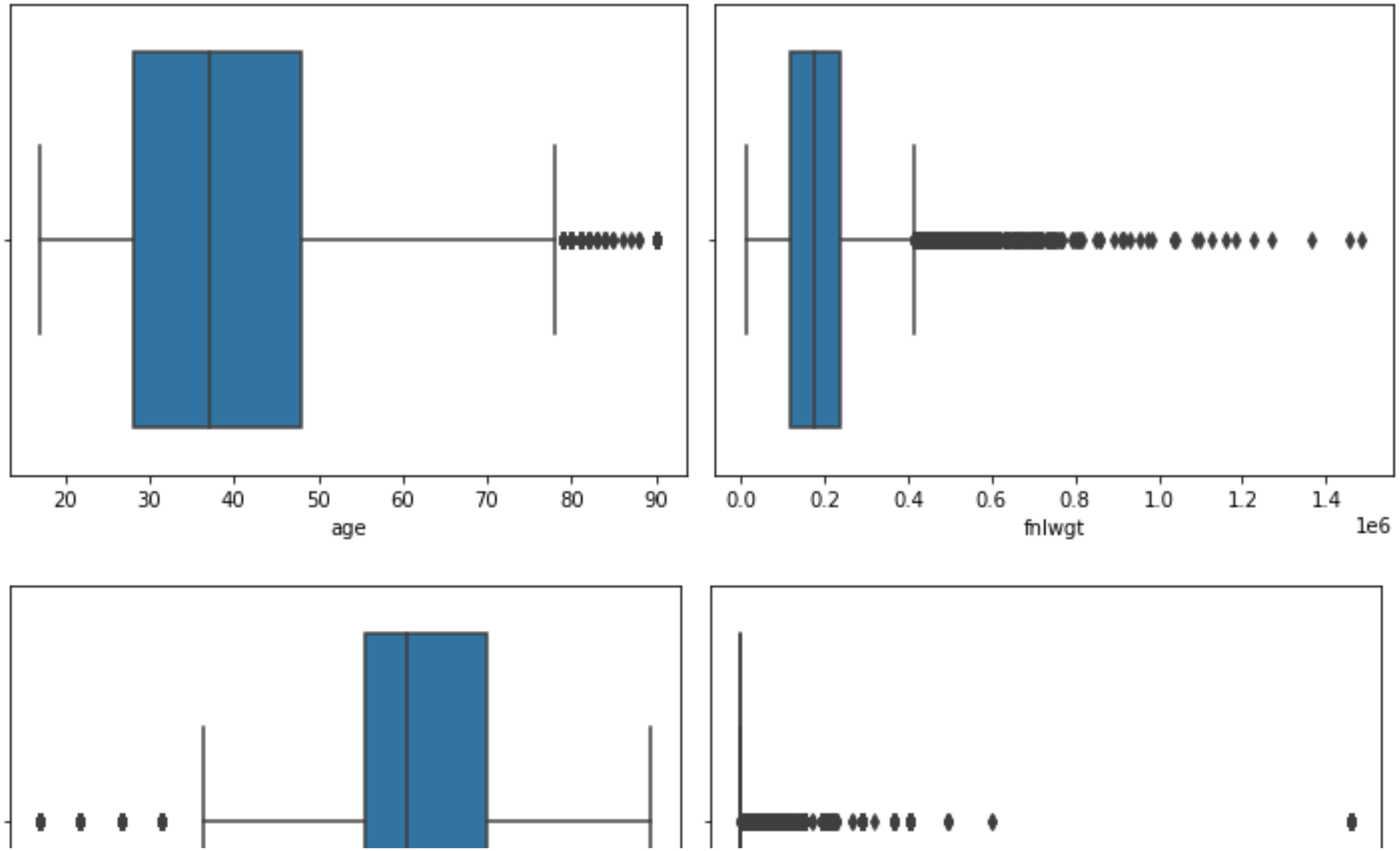
In [281]:
```python
plot_continuous_columns(num)
```

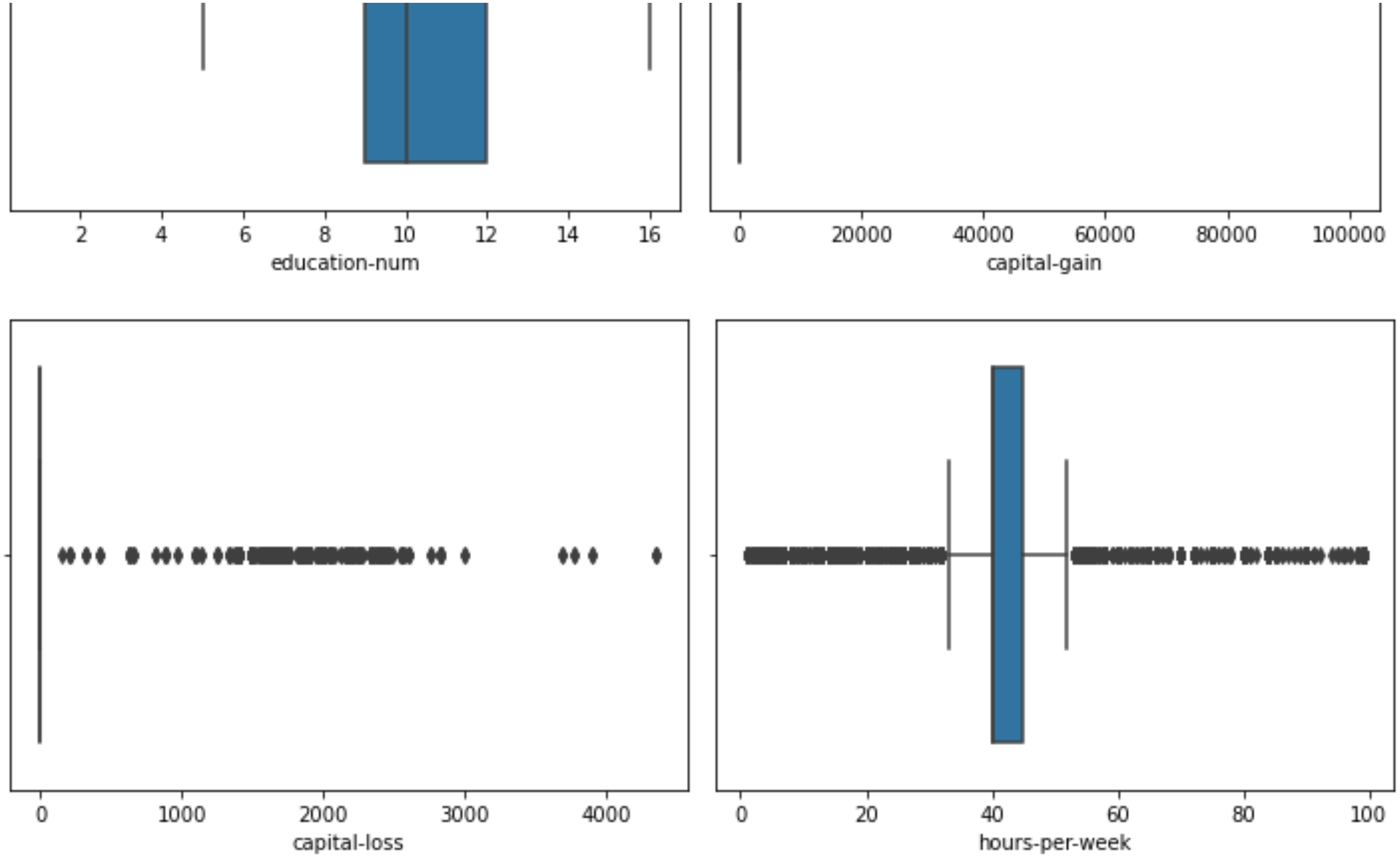

age
fnlwgt
1e6
education-num
capital-gain



In [282]:

```
plot_box_plots(num)
```





treating the outiers

In [283]:
```
from scipy.stats.mstats import winsorize
# Function to treat outliers
def treat_outliers(dataframe):
    cols = list(dataframe)
    for col in cols:
        if col in dataframe.select_dtypes(include=np.number).columns:
            dataframe[col] = winsorize(dataframe[col], limits=[0.05, 0.1],inclusive=(True, True))

    return dataframe
```

In [284]:
```
num=treat_outliers(num)
```

In [285]:
```
data.income.unique()[0]
```

Out[285]:
```
' <=50K'
```

In [286]:
```
data['income'] = data['income'].map({data.income.unique()[0]:0,data.income.unique()[1]:1})
```

In [287]:
```
data.tail()
```

Out[287]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States

In [298]:
```
pd.crosstab(data['sex'],data['income'])/data.shape[0]*100
```

Out[298]:

income	0	1
sex		
Female	29.459459	3.621007
Male	46.458845	20.460688

clearly shows that male are highly employed and female employment rate is low

splitting into x and y

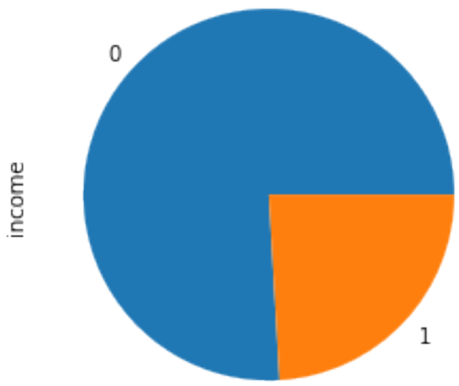
In [219]:
```
x=data.drop(['income'],axis=1)
y=data['income']
```

In [222]:
```
data['income'].value_counts()
```

Out[222]:
```
0    24719
1     7841
Name: income, dtype: int64
```

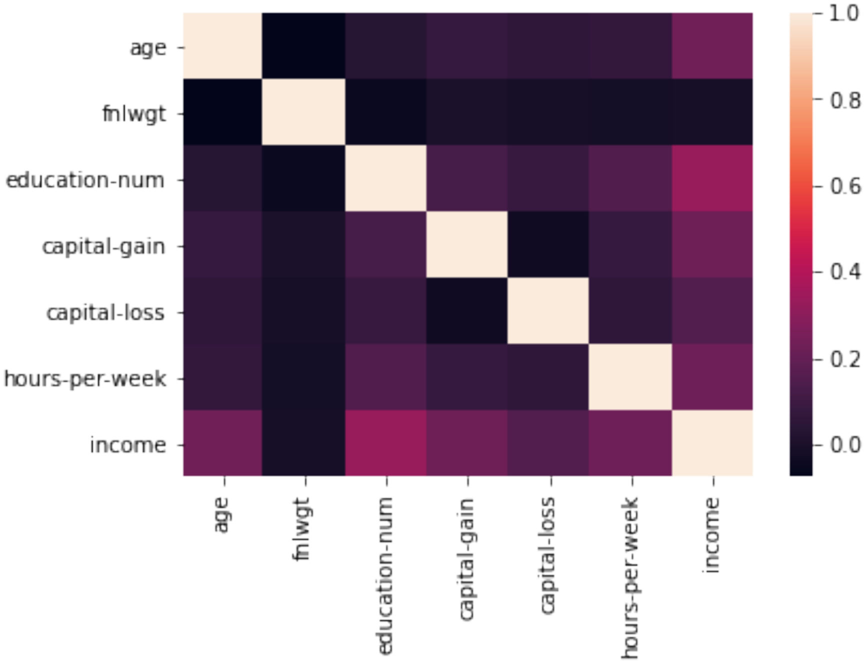
In [224]:
```
data['income'].value_counts().plot.pie()
```

Out[224]:
```
<AxesSubplot:ylabel='income'>
```



In [225]:
```
#pca,linear assuption : no multi correality
# from heatmap
sns.heatmap(data.corr())
```

Out[225]: <AxesSubplot:>



converting categorical into numerical data

In [239]:
```
x=pd.get_dummies(x)
```

In [240]:
```
x.head()
```

Out[240]:

rkclass_ ?	workclass_ Federal-gov	workclass_ Local-gov	workclass_ Never-worked	...	native-country_ Portugal	native-country_ Puerto-Rico	native-country_ Scotland	native-country_ South	native-country_ Taiwan	native-country_ Thailand	native-country_ Trinadad&Tobago	native-country_ United-States	nat cour Vieti
0	0	0	0	...	0	0	0	0	0	0	0	1	
0	0	0	0	...	0	0	0	0	0	0	0	1	
0	0	0	0	...	0	0	0	0	0	0	0	1	
0	0	0	0	...	0	0	0	0	0	0	0	0	
0	0	0	0	...	0	0	0	0	0	0	0	1	

Logistic Regression

In [241]:
```
from sklearn.model_selection import train_test_split
```

In [242]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [243]:
```
 from sklearn.linear_model import LogisticRegression
```

In [244]:
```
model=LogisticRegression()
```

In [245]:
```
model.fit(X_train,y_train)
```

Out[245]: LogisticRegression()

In [246]:
```
pred=model.predict(X_test)
```

In [255]:
```
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
```

In [256]:
```
confusion_matrix(pred,y_test)
```

Out[256]:
```
array([[4759, 1157],
       [ 159,  437]])
```

In [257]:
```
print(classification_report(pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.80      0.88      5916
           1       0.27      0.73      0.40       596

    accuracy                           0.80      6512
   macro avg       0.62      0.77      0.64      6512
weighted avg       0.90      0.80      0.83      6512
```

In [258]:
```
accuracy_score(pred,y_test)
```

Out[258]:
```
0.797911547911548
```

NSE

In [91]:
```
import pandas as pd
import numpy as np
```

In [92]:
```
import matplotlib.pyplot as plt
import pandas as pd

import datetime as dt
import numpy as np
import os
from sklearn.preprocessing import MinMaxScaler
```

In [93]:
```
dt=pd.read_csv('/Users/persie/Downloads/banknifty.csv')
```

In [94]:
```
dt.head()
```

Out[94]:

	index	date	time	open	high	low	close
0	BANKNIFTY	20121203	09:16	12125.70	12161.70	12125.70	12160.95
1	BANKNIFTY	20121203	09:17	12161.75	12164.80	12130.40	12130.40
2	BANKNIFTY	20121203	09:18	12126.85	12156.10	12126.85	12156.10
3	BANKNIFTY	20121203	09:19	12157.25	12164.75	12151.60	12164.20
4	BANKNIFTY	20121203	09:20	12162.80	12162.80	12148.20	12151.15

In [95]:
```
dt.describe()
```

Out[95]:

	date	open	high	low	close
count	3.675750e+05	367575.000000	367575.000000	367575.000000	367575.000000
mean	2.014401e+07	15078.023296	15082.498465	15073.480983	15077.993028
std	1.169302e+04	3184.438089	3185.213591	3183.628315	3184.411825
min	2.012110e+07	1405.050000	1407.050000	1404.600000	1405.200000
25%	2.013103e+07	12092.200000	12095.000000	12089.150000	12092.175000
50%	2.014110e+07	15526.100000	15531.200000	15521.400000	15525.950000
75%	2.015103e+07	17956.050000	17960.550000	17951.100000	17955.800000
max	2.016093e+07	20903.950000	20907.550000	20899.250000	20907.550000

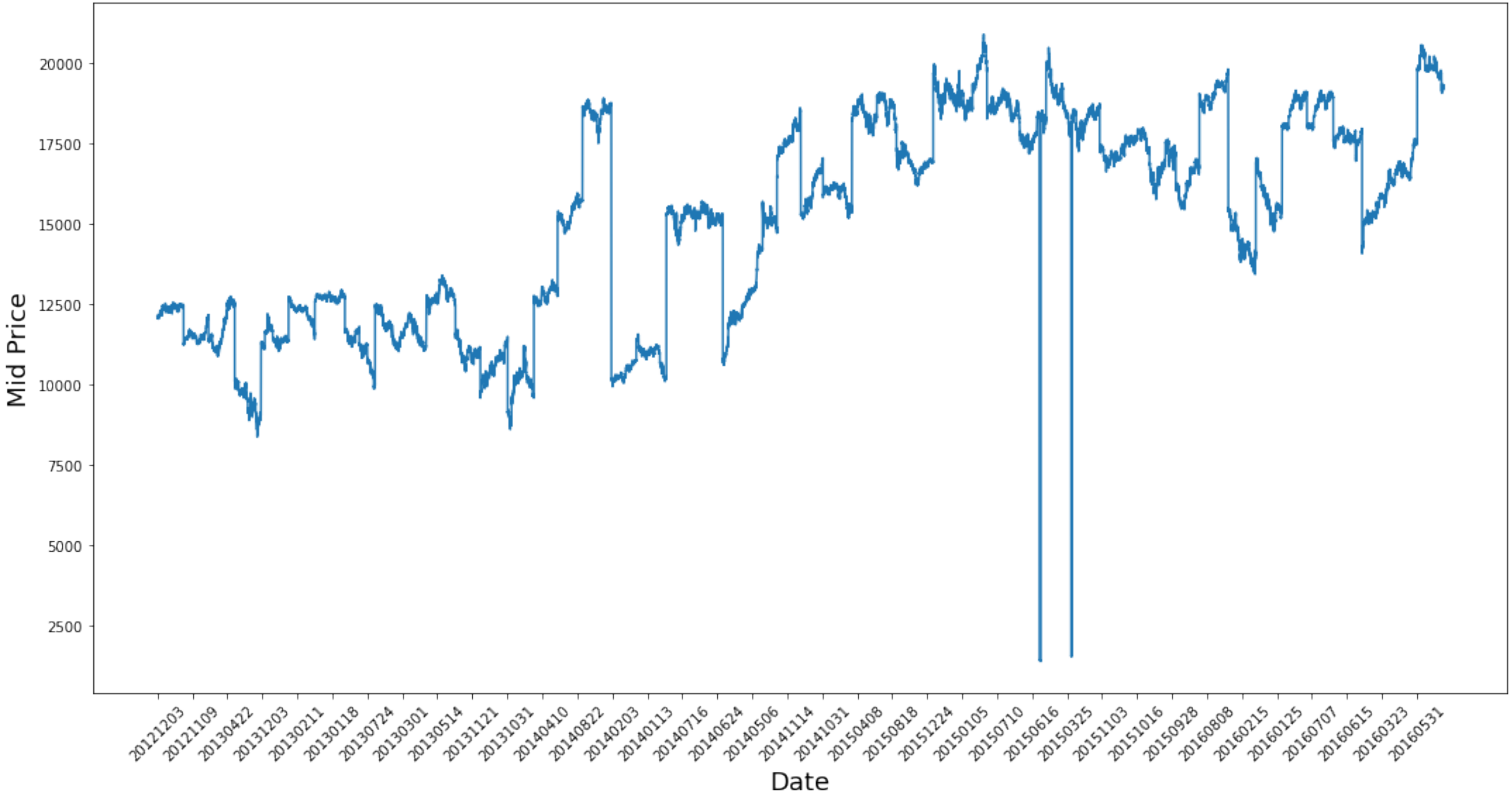
In [96]:
```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367575 entries, 0 to 367574
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   index   367575 non-null  object 
 1   date    367575 non-null  int64  
 2   time    367575 non-null  object 
 3   open    367575 non-null  float64
 4   high    367575 non-null  float64
 5   low     367575 non-null  float64
 6   close   367575 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 19.6+ MB
```

In [97]:
```
dt.isnull().sum()
```

Out[97]:
```
index    0
date     0
time     0
open     0
high     0
low      0
close    0
dtype: int64
```

In [98]:
```python
###### PLOTTING GRAPH ##############

plt.figure(figsize = (18,9))
plt.plot(range(dt.shape[0]),(dt['low']+dt['high'])/2.0)
# plt.plot(range(df.shape[0]),(df['volume']))
plt.xticks(range(0,dt.shape[0],10000),dt['date'].loc[::10000],rotation=45)
plt.xlabel('Date',fontsize=18)
plt.ylabel('Mid Price',fontsize=18)
plt.show()
```



Mid price range for every day

In [99]:
```
dt=dt.drop(['index','date','time'],axis=1)
```

In [100]:
```
y=dt["close"]
```

In [101]:
```
y.head()
```

Out[101]:
```
0    12160.95
1    12130.40
2    12156.10
3    12164.20
4    12151.15
Name: close, dtype: float64
```

In [102]:
```
x=dt.drop(['close'],axis=1)
```

In [103]:
```
x.head()
```

Out[103]:

	open	high	low
0	12125.70	12161.70	12125.70
1	12161.75	12164.80	12130.40
2	12126.85	12156.10	12126.85
3	12157.25	12164.75	12151.60
4	12162.80	12162.80	12148.20

Modelling

In [104]:
```
from sklearn.model_selection import train_test_split
```

In [105]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [106]:
```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(294060, 3) (73515, 3) (294060,) (73515,)
```

In [107]:
```
from sklearn.linear_model import LinearRegression
```

In [108]:
```
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[108]:
```
LinearRegression()
```

In [109]:
```
pred = model.predict(X_test)
```

In [110]:
```
from sklearn.metrics import r2_score

r2_score(y_test,pred)
```

Out[110]:
```
0.9999990415317453
```

accuracy = 99%