In this project, you are going to work on the The "Census Income" data set from the UCI Machine Learning Repository that contains the income information for over 48,000 individuals taken from the 1994 US census. For more details about this dataset, you can refer to the following link: https://archive.ics.uci.edu/ml/datasets/census+income

Problem Statement:

In this project, initially you need to preprocess the data and then develop an understanding of different features of the data by performing exploratory analysis and creating visualizations.Further, after having sufficient knowledge about the attributes you will perform a predictive task of classification to predict whether an individual makes over 50K a year or less,by using different Machine Learning Algorithms.

1. Data Preprocessing:

a) Replace all the missing values with NA.

b) Remove all the rows that contain NA values.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [176…
```
#Read the Census Income dataset
df = pd.read_csv(r'C:\Jagan\Personal\DS&AI Certification\Census_Income_Project\census-income.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	
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-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [4]:
```
df.describe()
```

Out[4]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
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	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [5]:
```
df.columns
```

Out[5]:
```
Index(['age', ' workclass', ' fnlwgt', ' education', ' education-num',
       ' marital-status', ' occupation', ' relationship', ' race', ' sex',
       ' capital-gain', ' capital-loss', ' hours-per-week', ' native-country',
       ' '],
      dtype='object')
```

In [6]:
```
df.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   hours-per-week  32561 non-null  int64
 13   native-country  32561 non-null  object
 14                   32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [14]:
```
df.isna().sum()
```

Out[14]:
```
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
                   0
dtype: int64
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
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
                  0
dtype: int64
```

In [18]:
```
df_columns = df.columns
df_columns
```

Out[18]:
```
Index(['age', ' workclass', ' fnlwgt', ' education', ' education-num',
       ' marital-status', ' occupation', ' relationship', ' race', ' sex',
       ' capital-gain', ' capital-loss', ' hours-per-week', ' native-country',
       ' '],
      dtype='object')
```

In [138…
```
#replacing space before the column names
df.columns = df.columns.str.replace(' ', '')
```

In [21]:
```
df.columns
```

Out[21]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', ''],
      dtype='object')
```

In [23]:
```
df.shape
```

Out[23]:
```
(32561, 15)
```

In [42]:
```
df.columns.values[14] = "YearlyIncome"
```

In [43]:
```
df.columns
```

Out[43]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

2. Data Manipulation:

a) Extract the “education” column and store it in “census_ed” .

b) Extract all the columns from “age” to “relationship” and store it in “census_seq”.

c) Extract the column number “5”, “8”, “11” and store it in “census_col”.

d) Extract all the male employees who work in state-gov and store it in “male_gov”.

e) Extract all the 39 year olds who either have a bachelor's degree or who are native of the United States and store the result in “census_us”.

f) Extract 200 random rows from the “census” data frame and store it in “census_200”.

g) Get the count of different levels of the “workclass” column.

h) Calculate the mean of the “capital.gain” column grouped according to “workclass”.

i) Create a separate dataframe with the details of males and females from the census data that has income more than 50,000.

j) Calculate the percentage of people from the United States who are private employees and earn less than 50,000 annually.

k) Calculate the percentage of married people in the census data.

l) Calculate the percentage of high school graduates earning more than 50,000 annually.

In [44]:
```
# Extract the “education” column and store it in “census_ed”
census_ed = df.loc[:,('education')]
```

In [45]:
```
type(census_ed)
```

Out[45]:
```
pandas.core.series.Series
```

In [46]:
```
census_ed.shape
```

Out[46]:
```
(32561,)
```

b) Extract all the columns from “age” to “relationship” and store it in “census_seq”

In [47]:
```
census_seq = df.loc[:,('age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship')]
```

In [48]:
```
census_seq.shape
```

Out[48]: (32561, 8)

c) Extract the column number “5”, “8”, “11” and store it in “census_col”

In [51]:
```
census_col = df.iloc[:,([5,8,11])]
```

In [53]:
```
census_col.columns
```

Out[53]: Index(['marital-status', 'race', 'capital-loss'], dtype='object')

In [54]:
```
census_col.shape
```

Out[54]: (32561, 3)

d) Extract all the male employees who work in state-gov and store it in “male_gov”.

In [58]:
```
df.sex.unique()
```

Out[58]: array([' Male', ' Female'], dtype=object)

In [59]:
```
df['workclass'].unique()
```

Out[59]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

In [66]:
```
male_gov = df.loc[(df['sex'] ==' Male') & (df['workclass'] == ' State-gov'),]
male_gov
```

Out[66]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
11	30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K
34	22	State-gov	311512	Some-college	10	Married-civ-spouse	Other-service	Husband	Black	Male	0	0	15	United-States	<=50K
48	41	State-gov	101603	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K
123	29	State-gov	267989	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	United-States	>50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32163	36	State-gov	135874	Bachelors	13	Married-civ-spouse	Sales	Husband	White	Male	0	0	40	United-States	<=50K
32241	45	State-gov	231013	Bachelors	13	Divorced	Protective-serv	Not-in-family	White	Male	0	0	40	United-States	<=50K
32321	54	State-gov	138852	HS-grad	9	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	40	United-States	<=50K
32324	42	State-gov	138162	Some-college	10	Divorced	Adm-clerical	Own-child	White	Male	0	0	40	United-States	<=50K
32360	58	State-gov	200316	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K

809 rows × 15 columns

In [67]:
```
male_gov.shape
```

Out[67]: (809, 15)

e) Extract all the 39 year olds who either have a bachelor's degree or who are native of the United States and store the result in “census_us”.

In [68]:
```
df.columns
```

Out[68]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

In [72]:
```
df['age'].unique()
```

Out[72]:
```
array([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], dtype=int64)
```

In [73]:
```
df['education'].unique()
```

Out[73]:
```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

In [74]:
```
df['native-country'].unique()
```

Out[74]:
```
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)
```

In [79]:
```
census_us = df.loc[(df['age'] == 39) & ((df['education'] == ' Bachelors') | (df['native-country'] == ' United-States')),]
```

In [80]:
```
census_us.shape
```

Out[80]: (759, 15)

In [81]:
```
census_us
```

Out[81]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
28	39	Private	367260	HS-grad	9	Divorced	Exec-managerial	Not-in-family	White	Male	0	0	80	United-States	<=50K

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
129	39	Private	365739	Some-college	10	Divorced	Craft-repair	Not-in-family	White	Male	0	0	40	United-States	<=50K
166	39	Federal-gov	235485	Assoc-acdm	12	Never-married	Exec-managerial	Not-in-family	White	Male	0	0	42	United-States	<=50K
320	39	Self-emp-not-inc	174308	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32146	39	Private	117381	Some-college	10	Divorced	Transport-moving	Not-in-family	White	Male	0	0	65	United-States	<=50K
32260	39	Federal-gov	232036	Some-college	10	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	40	United-States	>50K
32428	39	Federal-gov	110622	Bachelors	13	Married-civ-spouse	Adm-clerical	Wife	Asian-Pac-Islander	Female	0	0	40	Philippines	<=50K
32468	39	Self-emp-not-inc	193689	HS-grad	9	Never-married	Exec-managerial	Not-in-family	White	Male	0	0	65	United-States	<=50K
32545	39	Local-gov	111499	Assoc-acdm	12	Married-civ-spouse	Adm-clerical	Wife	White	Female	0	0	20	United-States	>50K

759 rows × 15 columns

f) Extract 200 random rows from the “census” data frame and store it in “census_200”.

In [83]:

```
df.shape
```

Out[83]:

```
(32561, 15)
```

In [88]:

```
import random

def Rand(start, end, num):
    res = []

    for j in range(num):
        res.append(random.randint(start, end))

    return res

census_200 = df.iloc[(Rand(0, 32561, 200))]
census_200
```

Out[88]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
21330	33	Private	178449	Some-college	10	Never-married	Adm-clerical	Not-in-family	White	Male	0	0	49	United-States	<=50K
32476	35	Private	30673	12th	8	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	84	United-States	<=50K
13135	24	State-gov	275421	Some-college	10	Never-married	Machine-op-inspct	Own-child	White	Female	0	0	40	United-States	<=50K
21957	27	State-gov	346406	Bachelors	13	Never-married	Prof-specialty	Unmarried	White	Male	0	0	50	United-States	<=50K
4373	35	Private	111128	10th	6	Never-married	Adm-clerical	Own-child	White	Male	0	0	40	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25065	37	Private	249208	Assoc-voc	11	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	48	United-States	>50K
9177	36	Private	114605	Assoc-voc	11	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	40	United-States	>50K
16829	44	State-gov	175696	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States	>50K
17795	67	Private	172756	1st-4th	2	Widowed	Machine-op-inspct	Not-in-family	White	Female	2062	0	34	Ecuador	<=50K
24079	34	Private	115858	HS-grad	9	Divorced	Adm-clerical	Own-child	White	Female	0	0	40	United-States	<=50K

200 rows × 15 columns

In [89]:

```
census_200.shape
```

Out[89]:

```
(200, 15)
```

In [91]:

```
census_200.head()
```

Out[91]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
21330	33	Private	178449	Some-college	10	Never-married	Adm-clerical	Not-in-family	White	Male	0	0	49	United-States	<=50K
32476	35	Private	30673	12th	8	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	84	United-States	<=50K
13135	24	State-gov	275421	Some-college	10	Never-married	Machine-op-inspct	Own-child	White	Female	0	0	40	United-States	<=50K
21957	27	State-gov	346406	Bachelors	13	Never-married	Prof-specialty	Unmarried	White	Male	0	0	50	United-States	<=50K
4373	35	Private	111128	10th	6	Never-married	Adm-clerical	Own-child	White	Male	0	0	40	United-States	<=50K

g) Get the count of different levels of the “workclass” column.

In [92]:

```
df.columns
```

Out[92]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

In [93]:

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

Out[93]:

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

In [99]:

```
df['workclass'].value_counts()
```

Out[99]:

```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

h) Calculate the mean of the “capital.gain” column grouped according to “workclass”.

In [101…

```
df.groupby('workclass')['capital-gain'].mean()
```

Out[101…

```
workclass
 ?                   606.795752
```

```
Federal-gov          833.232292
Local-gov            880.202580
Never-worked           0.000000
Private              889.217792
Self-emp-inc        4875.693548
Self-emp-not-inc    1886.061787
State-gov            701.699538
Without-pay          487.857143
Name: capital-gain, dtype: float64
```

i) Create a separate dataframe with the details of males and females from the census data that has income more than 50,000.

In [3]:

```python
columns = df.columns
columns
```

Out[3]:

```
Index(['age', ' workclass', ' fnlwgt', ' education', ' education-num',
       ' marital-status', ' occupation', ' relationship', ' race', ' sex',
       ' capital-gain', ' capital-loss', ' hours-per-week', ' native-country',
       ' '],
      dtype='object')
```

In [180…

```python
#replacing space before the column names
df.columns = df.columns.str.replace(' ', '')
```

In [9]:

```python
df.columns.values[14] = "YearlyIncome"
```

In [14]:

```python
df.columns
```

Out[14]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

In [177…

```python
# Rename single column by Index
df.rename(columns={df.columns[14]:  "YearlyIncome"},inplace=True)
print(df.columns)
```

```
Index(['age', ' workclass', ' fnlwgt', ' education', ' education-num',
       ' marital-status', ' occupation', ' relationship', ' race', ' sex',
       ' capital-gain', ' capital-loss', ' hours-per-week', ' native-country',
       'YearlyIncome'],
      dtype='object')
```

In [8]:

```python
df.sex.unique()
```

Out[8]:

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

In [17]:

```python
df['YearlyIncome'].value_counts()
```

Out[17]:

```
 <=50K    24720
 >50K      7841
Name: YearlyIncome, dtype: int64
```

In [49]:

```python
males_50k = df.loc[(df['sex'] == ' Male') & (df['YearlyIncome'] == ' >50K'),]
```

In [50]:

```python
males_50k.head()
```

Out[50]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K
10	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
11	30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K
14	40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	?	>50K

In [51]:

```python
females_50k = df.loc[(df['sex'] == ' Female') & (df['YearlyIncome'] == ' >50K'),]
```

In [52]:

```python
females_50k.head()
```

Out[52]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
19	43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	White	Female	0	0	45	United-States	>50K
52	47	Private	51835	Prof-school	15	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	1902	60	Honduras	>50K
67	53	Private	169846	HS-grad	9	Married-civ-spouse	Adm-clerical	Wife	White	Female	0	0	40	United-States	>50K
84	44	Private	343591	HS-grad	9	Divorced	Craft-repair	Not-in-family	White	Female	14344	0	40	United-States	>50K

In [53]:

```python
print("Count of records where sex = male and Income >=50k   : ", males_50k.shape[0])
print("Count of records where sex = female and Income >=50k : ", females_50k.shape[0])
```

```
Count of records where sex = male and Income >=50k   :  6662
Count of records where sex = female and Income >=50k :  1179
```

j) Calculate the percentage of people from the United States who are private employees and earn less than 50,000 annually.

In [34]:

```python
df.columns
```

Out[34]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

In [48]:

```python
df['native-country'].unique()
```

Out[48]:

```
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)
```

In [45]:

```
df.workclass.unique()
```

Out[45]:

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

In [47]:

```
df['YearlyIncome'].unique()
```

Out[47]:

```
array([' <=50K', ' >50K'], dtype=object)
```

In [55]:

```
df_us_under_50k_private = df.loc[(df['native-country'] == ' United-States') & (df['YearlyIncome'] == ' <=50K') & (df['workclass'] == ' Private'),]
df_us_under_50k_private.head()
```

Out[55]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
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-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
12	23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	<=50K
13	32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K

In [57]:

```
percentage_us_under_50k_private = (len(df_us_under_50k_private) * 100 )/ len(df)
print('percentage of US, under_50k income and private employed: ', percentage_us_under_50k_private)
```

```
percentage of US, under_50k income and private employed:  47.891649519363654
```

k) Calculate the percentage of married people in the census data.

In [59]:

```
df['marital-status'].unique()
```

Out[59]:

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

In [60]:

```
df['marital-status'].value_counts()
```

Out[60]:

```
 Married-civ-spouse       14976
 Never-married            10683
 Divorced                  4443
 Separated                 1025
 Widowed                    993
 Married-spouse-absent      418
 Married-AF-spouse           23
Name: marital-status, dtype: int64
```

In [70]:

```
married_count = df['marital-status'].value_counts()[' Married-civ-spouse'] + \
                df['marital-status'].value_counts()[' Married-spouse-absent'] +  \
                df['marital-status'].value_counts()[' Married-AF-spouse']
print("Percentage of married people : ", (married_count*100)/len(df))
```

```
Percentage of married people :  47.34805442093302
```

l) Calculate the percentage of high school graduates earning more than 50,000 annually.

In [71]:

```
df.education.unique()
```

Out[71]:

```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

In [72]:

```
df.education.value_counts()
```

Out[72]:

```
 HS-grad         10501
 Some-college     7291
 Bachelors        5355
 Masters          1723
 Assoc-voc        1382
 11th             1175
 Assoc-acdm       1067
 10th              933
 7th-8th           646
 Prof-school       576
 9th               514
 12th              433
 Doctorate         413
 5th-6th           333
 1st-4th           168
 Preschool          51
Name: education, dtype: int64
```

In [76]:

```
hs_grad_income_gt_50k = len (df.loc[(df['education'] == ' HS-grad') & (df['YearlyIncome'] == ' <=50K'),])
print("Highschool graduates with income >50k : ", hs_grad_income_gt_50k)
print("Total records count : ", len(df))
print("percentaage of Highschool graduates with income >50k : " ,(hs_grad_income_gt_50k *100 )/ len(df))
```

```
Highschool graduates with income >50k :  8826
Total records count :  32561
percentaage of Highschool graduates with income >50k :  27.106047111575197
```

3. Linear Regression:

a) Build a simple linear regression model as follows:

● Divide the dataset into training and test sets in 70:30 ratio.

● Build a linear model on the test set where the dependent variable is “hours.per.week” and the independent variable is “education.num”.

● Predict the values on the train set and find the error in prediction.

● Find the root-mean-square error (RMSE).

In [77]:
```
df.columns
```
Out[77]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```
In [87]:
```
x = df[['education-num']]
y = df[['hours-per-week']]
```
In [88]:
```
type(x), type(y)
```
Out[88]:
```
(pandas.core.frame.DataFrame, pandas.core.frame.DataFrame)
```
In [80]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```
In [81]:
```
lr=LinearRegression()
```
In [89]:
```
# Divide the dataset into training and test sets in 70:30 ratio.
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=1)
```
In [90]:
```
#fit the model on training data
lr.fit(x_train,y_train)
```
Out[90]:
```
LinearRegression()
```
In [91]:
```
#Predict the values on the train set
y_pred=lr.predict(x_test)
```
In [92]:
```
#find the error in prediction
error = y_test - y_pred
error
```
Out[92]:

	hours-per-week
9646	30.044869
709	-13.159243
7385	7.432533
16671	0.371349
21932	1.840757
...	...
29663	-1.832763
29310	0.371349
29661	-0.363355
19491	-1.098059
2861	5.514277

9769 rows × 1 columns

In [93]:
```
print('mean_squared_error :',mean_squared_error(y_test,y_pred))

print('root-mean-square error :',np.sqrt(mean_squared_error(y_test,y_pred)))
```
```
mean_squared_error : 147.15261838664162
root-mean-square error : 12.130647896408568
```

4. Logistic Regression:

a) Build a simple logistic regression model as follows:

● Divide the dataset into training and test sets in 65:35 ratio.

● Build a logistic regression model where the dependent variable is “X”(yearly income) and the independent variable is “occupation”.

● Predict the values on the test set.

● Build a confusion matrix and find the accuracy.

In [98]:
```
df.columns
```
Out[98]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```
In [94]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```
In [95]:
```
lo=LogisticRegression()
```
In [96]:
```
df[['occupation']].value_counts()
```
Out[96]:
```
occupation
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
```

```
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
dtype: int64
```

In [97]:
```
#occupation is indpendent

x=df['occupation'].replace('?','Prof-specialty')
x=pd.DataFrame(x)
```

In [99]:
```
df['YearlyIncome'].unique()
```

Out[99]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [100…
```
#income is dependent

y=df['YearlyIncome'].replace(' <=50K',0).replace(' >50K',1)
y=pd.DataFrame(y)
```

In [101…
```
x.head()
```

Out[101…

	occupation
0	Adm-clerical
1	Exec-managerial
2	Handlers-cleaners
3	Handlers-cleaners
4	Prof-specialty

In [102…
```
y.head()
```

Out[102…

	YearlyIncome
0	0
1	0
2	0
3	0
4	0

In [105…
```
le=LabelEncoder()
x=le.fit_transform(x)
```

```
C:\Users\577346744\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

In [106…
```
type(x)
```

Out[106…
```
numpy.ndarray
```

In [109…
```
x=pd.DataFrame(x)
```

In [110…
```
type(x)
```

Out[110…
```
pandas.core.frame.DataFrame
```

In [111…
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.35,random_state=1)
lo=LogisticRegression()
lo.fit(x_train,y_train)
y_pred=lo.predict(x_test)

print('confusion_matrix : ')
print(confusion_matrix(y_pred,y_test))
print('accuracy_score : ',accuracy_score(y_test,y_pred))
```

```
confusion_matrix :
[[8800 2597]
 [   0    0]]
accuracy_score :  0.7721330174607353
C:\Users\577346744\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

4 b)Build a multiple logistic regression model as follows:

● Divide the dataset into training and test sets in 80:20 ratio.

● Build a logistic regression model where the dependent variable is “X”(yearly income) and independent variables are “age”, “workclass”, and “education”.

● Predict the values on the test set.

● Build a confusion matrix and find the accuracy.

In [112…
```
df.head()
```

Out[112…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
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

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [113…

```
df.columns
```

Out[113…

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

In [114…

```
#independent

c=df[['age','workclass','education']]
```

In [115…

```
c.head()
```

Out[115…

	age	workclass	education
0	39	State-gov	Bachelors
1	50	Self-emp-not-inc	Bachelors
2	38	Private	HS-grad
3	53	Private	11th
4	28	Private	Bachelors

In [116…

```
c['workclass'].value_counts()
```

Out[116…

```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

In [117…

```
c['workclass']=c['workclass'].replace('?','Private')
```

```
<ipython-input-117-af2ecb2818e2>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  c['workclass']=c['workclass'].replace('?','Private')
```

In [118…

```
#columns transformation
c=c.apply(le.fit_transform)
c.head()
```

Out[118…

	age	workclass	education
0	22	7	9
1	33	6	9
2	21	4	11
3	36	4	1
4	11	4	9

In [119…

```
type(c)
```

Out[119…

```
pandas.core.frame.DataFrame
```

In [120…

```
x = c
```

In [125…

```
#income is dependent

y=df['YearlyIncome'].replace(' <=50K',0).replace(' >50K',1)
y=pd.DataFrame(y)
type(y)
```

Out[125…

```
pandas.core.frame.DataFrame
```

In [126…

```
y.head()
```

Out[126…

	YearlyIncome
0	0
1	0
2	0
3	0
4	0

In [127…

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=1)
lo=LogisticRegression()
lo.fit(x_train,y_train)
y_pred=lo.predict(x_test)

print('confusion_matrix : ')
print(confusion_matrix(y_pred,y_test))
print('accuracy_score : ',accuracy_score(y_test,y_pred))
```

```
confusion_matrix :
[[4905 1448]
 [ 121   39]]
accuracy_score :  0.7590971902349147
```

```
C:\Users\577346744\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

5. Decision Tree:

a) Build a decision tree model as follows:

● Divide the dataset into training and test sets in 70:30 ratio.

● Build a decision tree model where the dependent variable is “X”(Yearly Income) and the rest of the variables as independent variables.

● Predict the values on the test set.

● Build a confusion matrix and calculate the accuracy.

In [128…

```python
from sklearn.tree import DecisionTreeClassifier
```

In [181…

```python
df.head()
```

Out[181…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
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-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [182…

```python
df.columns
```

Out[182…

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

Feature Engineering

In [183…

```python
df.education.unique()
```

Out[183…

```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

In [184…

```python
# education Category
df.education = df.education.replace([' Preschool', ' 1st-4th', ' 5th-6th', ' 7th-8th', ' 9th',' 10th', ' 11th', ' 12th'], 'school')
df.education = df.education.replace(' HS-grad', 'highschool')
df.education = df.education.replace([' Assoc-voc', ' Assoc-acdm', ' Prof-school', ' Some-college'], 'higher')
df.education = df.education.replace(' Bachelors', 'undergrad')
df.education = df.education.replace(' Masters', 'grad')
df.education = df.education.replace(' Doctorate', 'doc')
```

In [185…

```python
df.education.unique()
```

Out[185…

```
array(['undergrad', 'highschool', 'school', 'grad', 'higher', 'doc'],
      dtype=object)
```

In [186…

```python
df['marital-status'].unique()
```

Out[186…

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

In [187…

```python
#martial status
df['marital-status']= df['marital-status'].replace([' Married-civ-spouse', ' Married-AF-spouse'], 'married')
df['marital-status']= df['marital-status'].replace([' Never-married'], 'not-married')
df['marital-status']= df['marital-status'].replace([' Divorced', ' Separated',' Widowed',' Married-spouse-absent'], 'other')
```

In [188…

```python
df['marital-status'].unique()
```

Out[188…

```
array(['not-married', 'married', 'other'], dtype=object)
```

In [189…

```python
df.columns
```

Out[189…

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

In [190…

```python
# Rename single column by Index
df.rename(columns={df.columns[14]:  "YearlyIncome"},inplace=True)
```

In [191…

```python
df.YearlyIncome.unique()
```

Out[191…

```
array([' <=50K', ' >50K'], dtype=object)
```

In [192…

```python
# income
df.income = df.YearlyIncome.replace(' <=50K', 0)
df.income = df.YearlyIncome.replace(' >50K', 1)
```

```
<ipython-input-192-d015b1d7e588>:2: UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  df.income = df.YearlyIncome.replace(' <=50K', 0)
```

In [193…

```python
# replacing ?
df['workclass']=df['workclass'].replace('?','Private')
df['occupation']=df['occupation'].replace('?','Prof-specialty')
df['native-country']=df['native-country'].replace('?','United-States')
```

In [194…

```python
df.head()
```

Out[194…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
0	39	State-gov	77516	undergrad	13	not-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	undergrad	13	married	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	highschool	9	other	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	school	7	married	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	undergrad	13	married	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [195…

```
df.columns
```

Out[195…

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'YearlyIncome'],
      dtype='object')
```

In [197…

```
#Column transform
backup=df.copy()
df=df.apply(le.fit_transform)
df.head()
```

Out[197…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	YearlyIncome
0	22	7	2671	5	12	1	1	1	4	1	25	0	39	39	0
1	33	6	2926	5	12	0	4	0	4	1	0	0	12	39	0
2	21	4	14086	3	8	2	6	1	4	1	0	0	39	39	0
3	36	4	15336	4	6	0	6	0	2	1	0	0	39	39	0
4	11	4	19355	5	12	0	10	5	2	0	0	0	39	5	0

In [198…

```
#independent
x=df.iloc[:,:-1]
#dependent
y=df.iloc[:,-1]
```

In [199…

```
type(x)
```

Out[199…

```
pandas.core.frame.DataFrame
```

In [200…

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=1)
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)

print('confusion_matrix : ')
print(confusion_matrix(y_pred,y_test))
print('accuracy_score : ',accuracy_score(y_test,y_pred))
```

```
confusion_matrix :
[[6547  830]
 [1003 1389]]
accuracy_score :  0.8123656464325929
```

6. Random Forest:

a) Build a random forest model as follows:

● Divide the dataset into training and test sets in 80:20 ratio.

● Build a random forest model where the dependent variable is “X”(Yearly Income) and the rest of the variables as independent variables and number of trees as 300.

● Predict values on the test set

● Build a confusion matrix and calculate the accuracy

In [201…

```
from sklearn.ensemble import RandomForestClassifier
```

In [202…

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=1)
rf=RandomForestClassifier(n_estimators=300)
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)

print('confusion_matrix : ')
print(confusion_matrix(y_pred,y_test))
print('accuracy_score : ',accuracy_score(y_test,y_pred))
```

```
confusion_matrix :
[[4658  523]
 [ 368  964]]
accuracy_score :  0.863196683555965
```

7. For this problem, use the population dataset, and perform the following:

1. EDA on the time series to find trends and seasonality.

2. Forecast the population on the given dataset for the next 6 months.

This problem was solved in Colab as prophet not working on Jupyter. Shared another file for that

In [203…

```
#Read the population dataset
popdata = pd.read_csv(r'C:\Jagan\Personal\DS&AI Certification\Census_Income_Project\popdata.csv')
```

In [204…

```
popdata.head()
```

Out[204...

	value	date
0	127299.0	1952-01-01
1	127517.0	1952-02-01
2	127721.0	1952-03-01
3	127933.0	1952-04-01
4	128130.0	1952-05-01

In [205...

```
popdata.columns
```

Out[205...

```
Index(['value', 'date'], dtype='object')
```

In [206...

```
popdata.describe()
```

Out[206...

	value
count	816.000000
mean	214837.767826
std	50519.140567
min	127299.000000
25%	172715.250000
50%	210547.500000
75%	260354.250000
max	301299.946000

In [207...

```
popdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   value   816 non-null    float64
 1   date    816 non-null    object 
dtypes: float64(1), object(1)
memory usage: 12.9+ KB
```

In [209...

```
popdata.isnull().sum()
```

Out[209...

```
value    0
date     0
dtype: int64
```

In [210...

```
from datetime import datetime
```

In [211...

```
popdata['Value']=popdata['value']
popdata=popdata.drop(columns=['value'])
popdata.head()
```

Out[211...

	date	Value
0	1952-01-01	127299.0
1	1952-02-01	127517.0
2	1952-03-01	127721.0
3	1952-04-01	127933.0
4	1952-05-01	128130.0

In [212...

```
popdata.columns=['ds','y']
```

In [213...

```
popdata.head()
```

Out[213...

	ds	y
0	1952-01-01	127299.0
1	1952-02-01	127517.0
2	1952-03-01	127721.0
3	1952-04-01	127933.0
4	1952-05-01	128130.0

In [227...

```
from fbprophet import Prophet
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-227-f503e9c6cf11> in <module>
----> 1 from fbprophet import Prophet

ModuleNotFoundError: No module named 'fbprophet'
```

In [226...

```
pip install --upgrade setuptools
```

```
Requirement already satisfied: setuptools in c:\users\577346744\anaconda3\lib\site-packages (52.0.0.post20210125)
Collecting setuptools
  Downloading setuptools-62.3.2-py3-none-any.whl (1.2 MB)
Installing collected packages: setuptools
  Attempting uninstall: setuptools
    Found existing installation: setuptools 52.0.0.post20210125
    Uninstalling setuptools-52.0.0.post20210125:
      Successfully uninstalled setuptools-52.0.0.post20210125
Successfully installed setuptools-62.3.2
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
spyder 4.2.5 requires pyqt5<5.13, which is not installed.
spyder 4.2.5 requires pyqtwebengine<5.13, which is not installed.
conda-repo-cli 1.0.4 requires pathlib, which is not installed.
```

In [222...

```
pip install fbprophet
```

```
Collecting fbprophet
  Using cached fbprophet-0.7.1.tar.gz (64 kB)
Requirement already satisfied: Cython>=0.22 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (0.29.23)
Requirement already satisfied: cmdstanpy==0.9.5 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (0.9.5)
Requirement already satisfied: pystan>=2.14 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (2.19.1.1)
Requirement already satisfied: numpy>=1.15.4 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (1.20.1)
Requirement already satisfied: pandas>=1.0.4 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (1.2.4)
```

```
Requirement already satisfied: matplotlib>=2.0.0 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (3.3.4)
Requirement already satisfied: LunarCalendar>=0.0.9 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (0.0.9)
Requirement already satisfied: convertdate>=2.1.2 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (2.4.0)
Requirement already satisfied: holidays>=0.10.2 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (0.13)
Requirement already satisfied: setuptools-git>=1.2 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (1.2)
Requirement already satisfied: python-dateutil>=2.8.0 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (2.8.1)
Requirement already satisfied: tqdm>=4.36.1 in c:\users\577346744\anaconda3\lib\site-packages (from fbprophet) (4.59.0)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in c:\users\577346744\anaconda3\lib\site-packages (from convertdate>=2.1.2->fbprophet) (0.5.11)
Requirement already satisfied: hijri-converter in c:\users\577346744\anaconda3\lib\site-packages (from holidays>=0.10.2->fbprophet) (2.2.4)
Requirement already satisfied: korean-lunar-calendar in c:\users\577346744\anaconda3\lib\site-packages (from holidays>=0.10.2->fbprophet) (0.2.1)
Requirement already satisfied: pytz in c:\users\577346744\anaconda3\lib\site-packages (from LunarCalendar>=0.0.9->fbprophet) (2021.1)
Requirement already satisfied: ephem>=3.7.5.3 in c:\users\577346744\anaconda3\lib\site-packages (from LunarCalendar>=0.0.9->fbprophet) (4.1.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\577346744\anaconda3\lib\site-packages (from matplotlib>=2.0.0->fbprophet) (8.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\577346744\anaconda3\lib\site-packages (from matplotlib>=2.0.0->fbprophet) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\577346744\anaconda3\lib\site-packages (from matplotlib>=2.0.0->fbprophet) (2.4.7)
Requirement already satisfied: cycler>=0.10 in c:\users\577346744\anaconda3\lib\site-packages (from matplotlib>=2.0.0->fbprophet) (0.10.0)
Requirement already satisfied: six in c:\users\577346744\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib>=2.0.0->fbprophet) (1.15.0)
Building wheels for collected packages: fbprophet
  Building wheel for fbprophet (setup.py): started
  Building wheel for fbprophet (setup.py): finished with status 'error'Note: you may need to restart the kernel to use updated packages.
  Running setup.py clean for fbprophet
Failed to build fbprophet
Installing collected packages: fbprophet
    Running setup.py install for fbprophet: started
    Running setup.py install for fbprophet: finished with status 'error'
```

```
  ERROR: Command errored out with exit status 1:
   command: 'C:\Users\577346744\Anaconda3\python.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'"'C:\\Users\\577346744\\AppData\\Local\\Temp\\pip-install-udhyfcac\\fbprophet_1bf20ddd40624185b7f663
4c876e0b28\\setup.py'"'"'; __file__='"'"'C:\\Users\\577346744\\AppData\\Local\\Temp\\pip-install-udhyfcac\\fbprophet_1bf20ddd40624185b7f6634c876e0b28\\setup.py'"'"';f=getattr(tokenize, '"'"'open'"'"', open)(__fil
e__);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' bdist_wheel -d 'C:\Users\577346744\AppData\Local\Temp\pip-wheel-42ek75c5'
       cwd: C:\Users\577346744\AppData\Local\Temp\pip-install-udhyfcac\fbprophet_1bf20ddd40624185b7f6634c876e0b28\
  Complete output (11 lines):
  running bdist_wheel
  running build
  running build_py
  creating build
  creating build\lib
  creating build\lib\fbprophet
  creating build\lib\fbprophet\stan_model
  Importing plotly failed. Interactive plots will not work.
  INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_f5236004a3fd5b8429270d00efcc0cf9 NOW.
  WARNING:pystan:MSVC compiler is not supported
  error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
  ----------------------------------------
  ERROR: Failed building wheel for fbprophet
    ERROR: Command errored out with exit status 1:
     command: 'C:\Users\577346744\Anaconda3\python.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'"'C:\\Users\\577346744\\AppData\\Local\\Temp\\pip-install-udhyfcac\\fbprophet_1bf20ddd40624185b7f6
634c876e0b28\\setup.py'"'"'; __file__='"'"'C:\\Users\\577346744\\AppData\\Local\\Temp\\pip-install-udhyfcac\\fbprophet_1bf20ddd40624185b7f6634c876e0b28\\setup.py'"'"';f=getattr(tokenize, '"'"'open'"'"', open)(__f
ile__);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' install --record 'C:\Users\577346744\AppData\Local\Temp\pip-record-vkrc35ng\install-record.txt'
--single-version-externally-managed --compile --install-headers 'C:\Users\577346744\Anaconda3\Include\fbprophet'
         cwd: C:\Users\577346744\AppData\Local\Temp\pip-install-udhyfcac\fbprophet_1bf20ddd40624185b7f6634c876e0b28\
    Complete output (11 lines):
    running install
    running build
    running build_py
    creating build
    creating build\lib
    creating build\lib\fbprophet
    creating build\lib\fbprophet\stan_model
    Importing plotly failed. Interactive plots will not work.
    INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_f5236004a3fd5b8429270d00efcc0cf9 NOW.
    WARNING:pystan:MSVC compiler is not supported
    error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
    ----------------------------------------
ERROR: Command errored out with exit status 1: 'C:\Users\577346744\Anaconda3\python.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'"'C:\\Users\\577346744\\AppData\\Local\\Temp\\pip-install-udhyfca
c\\fbprophet_1bf20ddd40624185b7f6634c876e0b28\\setup.py'"'"'; __file__='"'"'C:\\Users\\577346744\\AppData\\Local\\Temp\\pip-install-udhyfcac\\fbprophet_1bf20ddd40624185b7f6634c876e0b28\\setup.py'"'"';f=getattr(to
kenize, '"'"'open'"'"', open)(__file__);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' install --record 'C:\Users\577346744\AppData\Local\Temp\pip-rec
ord-vkrc35ng\install-record.txt' --single-version-externally-managed --compile --install-headers 'C:\Users\577346744\Anaconda3\Include\fbprophet' Check the logs for full command output.
```

In [224…

```
pip install cryptography
```

```
Requirement already satisfied: cryptography in c:\users\577346744\anaconda3\lib\site-packages (3.4.7)
Requirement already satisfied: cffi>=1.12 in c:\users\577346744\anaconda3\lib\site-packages (from cryptography) (1.14.5)
Requirement already satisfied: pycparser in c:\users\577346744\anaconda3\lib\site-packages (from cffi>=1.12->cryptography) (2.20)
Note: you may need to restart the kernel to use updated packages.
```

In [214…

```
#model fit
model=Prophet()
model.fit(popdata)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-214-60e64c733252> in <module>
      1 #model fit
----> 2 model=Prophet()
      3 model.fit(popdata)

NameError: name 'Prophet' is not defined
```