```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

+ Code + Text

```
data = pd.read_csv(('/content/adult.data'),header=None,delimiter=' *, *',engine='python')
```

```
data
```

	0	1	2	3	4	5	6	7	8	9
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female
...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-	12	Married-civ-	Tech-	Wife	White	Female

```
data.columns=['age','workclass','fnlwgt','education','education_num','marital_status',
              'occupation','relationship','race','sex','capital_gain','capital_loss',
              'hours_per_week','native_country','salary']
```

```
data.head()
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationsl
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-far
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husbi
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-far
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husbi
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	V

```
data.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
hours_per_week    0
native_country    0
```

```
salary            0
dtype: int64
```

```
data.dtypes
```

```
age                int64
workclass         object
fnlwgt             int64
education         object
education_num      int64
marital_status    object
occupation        object
relationship      object
race              object
sex               object
capital_gain       int64
capital_loss       int64
hours_per_week     int64
native_country    object
salary            object
dtype: object
```

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

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

```
data['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)
```

```
data['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)
```

```
for value in data.columns:
    print(data[value].unique())
```

```
[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]
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov' '?'
 'Self-emp-inc' 'Without-pay' 'Never-worked']
[ 77516  83311 215646 ...  34066  84661 257302]
['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']
[13  9  7 14  5 10 12 11  4 16 15  3  6  2  1  8]
['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']
[ 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]
[   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
 2231 1844  810 2824 2559 2057 1974  974 2149 1825 1735 1258 2129 2603
 2282  323 4356 2246 1617 1648 2489 3770 1755 3683 2267 2080 2457  155
 3900 2201 1944 2467 2163 2754 2472 1411]
[40 13 16 45 50 80 30 35 60 20 52 44 15 25 38 43 55 48 58 32 70  2 22 56
 41 28 36 24 46 42 12 65  1 10 34 75 98 33 54  8  6 64 19 18 72  5  9 47
 37 21 26 14  4 59  7 99 53 39 62 57 78 90 66 11 49 84  3 17 68 27 85 31
 51 77 63 23 87 88 73 89 97 94 29 96 67 82 86 91 81 76 92 61 74 95]
['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']
```

```
for value in data.columns:
    print(value,":", sum(data[value] == '?'))
```

```
age : 0
workclass : 1836
fnlwgt : 0
education : 0
education_num : 0
marital_status : 0
occupation : 1843
relationship : 0
race : 0
sex : 0
capital_gain : 0
capital_loss : 0
hours_per_week : 0
native_country : 583
salary : 0
```

```
data1=data.describe(include='all')
```

```
for value in ['workclass','occupation','native_country']:
    data[value].replace('?',data1[value][2],inplace=True)
```

```
data1.dtypes
```

```
age               float64
workclass          object
fnlwgt            float64
education          object
education_num     float64
marital_status     object
occupation         object
relationship       object
race               object
sex                object
capital_gain      float64
capital_loss      float64
hours_per_week    float64
native_country     object
salary             object
dtype: object
```

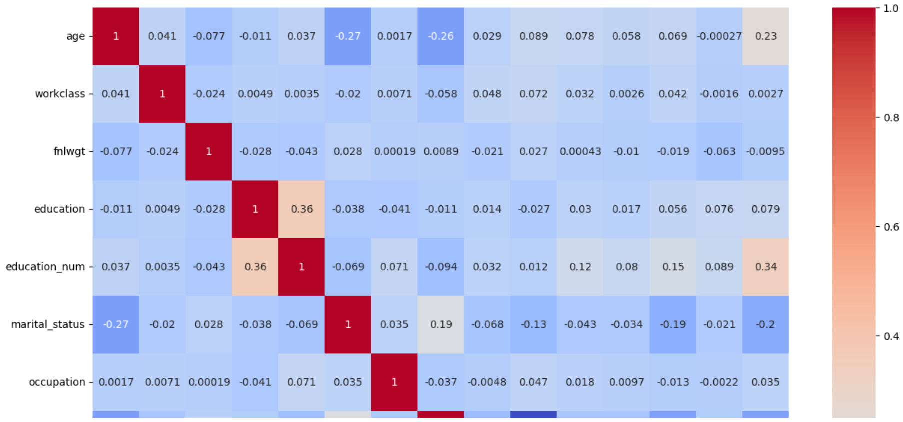
```
from sklearn.preprocessing import LabelEncoder
```

```
le1=LabelEncoder()
data.workclass=le1.fit_transform(data.workclass)

le2=LabelEncoder()
data.education=le2.fit_transform(data.education)

le3=LabelEncoder()
```

```
data.marital_status=le3.fit_transform(data.marital_status)

le4=LabelEncoder()
data.occupation=le4.fit_transform(data.occupation)

le5=LabelEncoder()
data.relationship=le5.fit_transform(data.relationship)

le6=LabelEncoder()
data.race=le6.fit_transform(data.race)

le7=LabelEncoder()
data.sex=le7.fit_transform(data.sex)

le8=LabelEncoder()
data.native_country=le8.fit_transform(data.native_country)

le9=LabelEncoder()
data.salary=le9.fit_transform(data.salary)
```

```
plt.figure(figsize=(15,15))
cor=data.corr()
sns.heatmap(cor,annot=True,cmap='coolwarm')
```

```
<Axes: >
```




```
data2=data.drop(['fnlwgt'],axis=1)
```

```
ip=data2.drop(['salary'],axis=1)
op=data['salary']
```

```
from keras.utils.np_utils import to_categorical
ip.workclass = to_categorical(ip.workclass)
ip.marital_status = to_categorical(ip.marital_status)
ip.education = to_categorical(ip.education)
ip.relationship = to_categorical(ip.relationship)
ip.race = to_categorical(ip.race)
ip.native_country = to_categorical(ip.native_country)
ip.sex = to_categorical(ip.sex)
```


```
ip.shape
```

```
(32561, 13)
```

```
op =to_categorical(op,2)
```

```
from sklearn.model_selection import train_test_split
xtr,xts,ytr,yts= train_test_split(ip,op,test_size=0.2)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(xtr)
sc.fit(xts)
xtr =sc.transform(xtr)
xts =sc.transform(xts)
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
```

```
model = Sequential()
model.add(Dense(100,input_dim=13,activation='sigmoid'))
model.add(Dense(60,activation='sigmoid'))
model.add(Dense(60,activation='sigmoid'))
model.add(Dense(2,activation='sigmoid'))
model.compile(Adam(lr=0.03),loss='binary_crossentropy',metrics=['accuracy'])
```

```
print(model.summary())
```

```
h = model.fit(xtr,ytr,epochs=60,validation_data=(xts,yts))
```

```
Epoch 35/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3841 - accuracy: 0.8221 - val_loss: 0.3786 - val_accuracy: 0.8262
Epoch 36/60
814/814 [==============================] - 3s 4ms/step - loss: 0.3835 - accuracy: 0.8240 - val_loss: 0.3711 - val_accuracy: 0.8268
Epoch 37/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3842 - accuracy: 0.8235 - val_loss: 0.3756 - val_accuracy: 0.8204
Epoch 38/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3851 - accuracy: 0.8192 - val_loss: 0.3721 - val_accuracy: 0.8187
Epoch 39/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3822 - accuracy: 0.8240 - val_loss: 0.3775 - val_accuracy: 0.8302
Epoch 40/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3823 - accuracy: 0.8264 - val_loss: 0.3713 - val_accuracy: 0.8282
Epoch 41/60
814/814 [==============================] - 3s 4ms/step - loss: 0.3842 - accuracy: 0.8248 - val_loss: 0.3730 - val_accuracy: 0.8190
Epoch 42/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3841 - accuracy: 0.8217 - val_loss: 0.3725 - val_accuracy: 0.8299
Epoch 43/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3816 - accuracy: 0.8247 - val_loss: 0.3830 - val_accuracy: 0.8217
Epoch 44/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3824 - accuracy: 0.8271 - val_loss: 0.3809 - val_accuracy: 0.8197
Epoch 45/60
814/814 [==============================] - 3s 4ms/step - loss: 0.3841 - accuracy: 0.8232 - val_loss: 0.3732 - val_accuracy: 0.8299
Epoch 46/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3811 - accuracy: 0.8254 - val_loss: 0.3713 - val_accuracy: 0.8306
Epoch 47/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3803 - accuracy: 0.8269 - val_loss: 0.3702 - val_accuracy: 0.8328
Epoch 48/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3821 - accuracy: 0.8245 - val_loss: 0.3734 - val_accuracy: 0.8303
Epoch 49/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3831 - accuracy: 0.8259 - val_loss: 0.3696 - val_accuracy: 0.8328
Epoch 50/60
814/814 [==============================] - 3s 4ms/step - loss: 0.3842 - accuracy: 0.8261 - val_loss: 0.3721 - val_accuracy: 0.8294
Epoch 51/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3830 - accuracy: 0.8250 - val_loss: 0.3698 - val_accuracy: 0.8302
Epoch 52/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3821 - accuracy: 0.8270 - val_loss: 0.3847 - val_accuracy: 0.8310
Epoch 53/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3834 - accuracy: 0.8269 - val_loss: 0.3794 - val_accuracy: 0.8340
Epoch 54/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3834 - accuracy: 0.8264 - val_loss: 0.3862 - val_accuracy: 0.8305
Epoch 55/60
814/814 [==============================] - 3s 4ms/step - loss: 0.3819 - accuracy: 0.8269 - val_loss: 0.3693 - val_accuracy: 0.8244
Epoch 56/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3820 - accuracy: 0.8264 - val_loss: 0.3788 - val_accuracy: 0.8300
Epoch 57/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3824 - accuracy: 0.8262 - val_loss: 0.3836 - val_accuracy: 0.8265
Epoch 58/60
814/814 [==============================] - 2s 3ms/step - loss: 0.3825 - accuracy: 0.8251 - val_loss: 0.3804 - val_accuracy: 0.8263
Epoch 59/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3835 - accuracy: 0.8240 - val_loss: 0.3800 - val_accuracy: 0.8300
Epoch 60/60
814/814 [==============================] - 3s 3ms/step - loss: 0.3834 - accuracy: 0.8266 - val_loss: 0.3814 - val_accuracy: 0.8254
```

```
yp = model.predict(xts)
```

```
204/204 [==============================] - 0s 2ms/step
```

```
yp1 = np.argmax(yp,axis=1)
```

```
yp1
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score
recall= recall_score(yp1,np.argmax(yts,axis=1))
confusion = confusion_matrix(yp1,np.argmax(yts,axis=1))
print(recall)
print(confusion)
```

```
0.7939914163090128
[[4821  993]
 [ 144  555]]
```