Income_Classification

September 1, 2022

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv("income_evaluation.csv")
```

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

[3]:

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

[4]:
```
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   income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[5]:
```
for i in df.columns:
    print(df[i].unique())
```

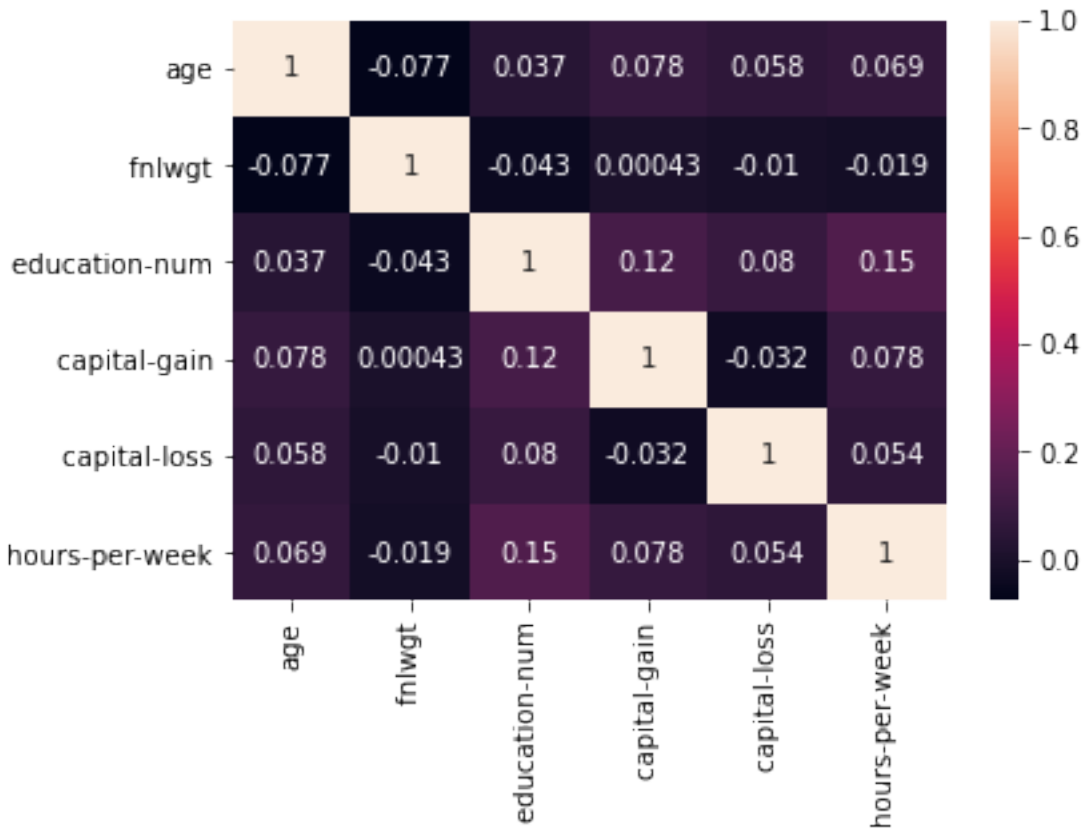
```
[39 50 38 53 28 37 49 52 31 42 30 23 32 40 34 25 43 54 35 59 56 19 20 45
 22 48 21 24 57 44 41 29 18 47 46 36 79 27 67 33 76 17 55 61 70 64 71 68
 66 51 58 26 60 90 75 65 77 62 63 80 72 74 69 73 81 78 88 82 83 84 85 86
 87]
[' State-gov' ' Self-emp-not-inc' ' Private' ' Federal-gov' ' Local-gov'
 ' ?' ' 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']
```

[6]:
```
import seaborn as sns
```

[7]:
```
sns.heatmap(df.corr(),annot=True)
```

[7]:
```
<AxesSubplot:>
```



[8]:
```
num_df = df.select_dtypes(include=["int64"])
```

[9]:
```
num_df
```

[9]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	
3	53	234721	7	0	0	
4	28	338409	13	0	0	
…	…	…	…	…	…	
32556	27	257302	12	0	0	
32557	40	154374	9	0	0	
32558	58	151910	9	0	0	
32559	22	201490	9	0	0	
32560	52	287927	9	15024	0	

```
2                   40
3                   40
4                   40
…                   …
32556               38
32557               40
32558               40
32559               20
32560               40

[32561 rows x 6 columns]
```

[10]:
```
cat_df = df.select_dtypes(exclude=["int64"])
```

[11]:
```
cat_df
```

[11]:
```
                workclass    education       marital-status  \
0               State-gov    Bachelors        Never-married
1       Self-emp-not-inc    Bachelors   Married-civ-spouse
2                 Private      HS-grad             Divorced
3                 Private         11th   Married-civ-spouse
4                 Private    Bachelors   Married-civ-spouse
…                   …          …                 …
32556             Private   Assoc-acdm   Married-civ-spouse
32557             Private      HS-grad   Married-civ-spouse
32558             Private      HS-grad              Widowed
32559             Private      HS-grad        Never-married
32560       Self-emp-inc      HS-grad   Married-civ-spouse

                occupation    relationship    race      sex  native-country  \
0            Adm-clerical   Not-in-family   White     Male   United-States
1          Exec-managerial         Husband   White     Male   United-States
2        Handlers-cleaners   Not-in-family   White     Male   United-States
3        Handlers-cleaners         Husband   Black     Male   United-States
4           Prof-specialty            Wife   Black   Female            Cuba
…                     …             …      …      …              …
32556         Tech-support            Wife   White   Female   United-States
32557   Machine-op-inspct         Husband   White     Male   United-States
32558        Adm-clerical       Unmarried   White   Female   United-States
32559        Adm-clerical       Own-child   White     Male   United-States
32560      Exec-managerial            Wife   White   Female   United-States

       income
0       <=50K
1       <=50K
2       <=50K
3       <=50K
```

```
4       <=50K
…       …
32556   <=50K
32557    >50K
32558   <=50K
32559   <=50K
32560    >50K

[32561 rows x 9 columns]
```

[12]:
```
from sklearn.preprocessing import LabelEncoder
```

[13]:
```
encoder = LabelEncoder()
```

[14]:
```
type(cat_df)
```

[14]:
```
pandas.core.frame.DataFrame
```

[15]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

[16]:
```
num_df.columns
```

[16]:
```
Index(['age', ' fnlwgt', ' education-num', ' capital-gain', ' capital-loss',
       ' hours-per-week'],
      dtype='object')
```

[17]:
```
for i in cat_df.columns:
    df[i] = encoder.fit_transform(cat_df[i])
```

[18]:
```
df[['age', ' fnlwgt', ' education-num', ' capital-gain', ' capital-loss',
       ' hours-per-week']] = scaler.fit_transform(df[['age', ' fnlwgt', '␣
↪education-num', ' capital-gain', ' capital-loss',
       ' hours-per-week']])
```

[19]:
```
df
```

[19]:
```
            age   workclass    fnlwgt   education   education-num  \
0      0.030671           7 -1.063611           9        1.134739
1      0.837109           6 -1.008707           9        1.134739
2     -0.042642           4  0.245079          11       -0.420060
3      1.057047           4  0.425801           1       -1.197459
4     -0.775768           4  1.408176           9        1.134739
…           …         …        …           …              …
32556 -0.849080           4  0.639741           7        0.746039
32557  0.103983           4 -0.335433          11       -0.420060
32558  1.423610           4 -0.358777          11       -0.420060
```

```
32559 -1.215643           4  0.110960          11       -0.420060
32560  0.983734           5  0.929893          11       -0.420060

       marital-status   occupation   relationship   race   sex  \
0                   4            1              1      4     1
1                   2            4              0      4     1
2                   0            6              1      4     1
3                   2            6              0      2     1
4                   2           10              5      2     0
…                 …            …            …     …   …
32556               2           13              5      4     0
32557               2            7              0      4     1
32558               6            1              4      4     0
32559               4            1              3      4     1
32560               2            4              5      4     0

       capital-gain   capital-loss   hours-per-week   native-country   income
0          0.148453      -0.21666        -0.035429               39        0
1         -0.145920      -0.21666        -2.222153               39        0
2         -0.145920      -0.21666        -0.035429               39        0
3         -0.145920      -0.21666        -0.035429               39        0
4         -0.145920      -0.21666        -0.035429                5        0
…               …            …               …              …       …
32556     -0.145920      -0.21666        -0.197409               39        0
32557     -0.145920      -0.21666        -0.035429               39        1
32558     -0.145920      -0.21666        -0.035429               39        0
32559     -0.145920      -0.21666        -1.655225               39        0
32560      1.888424      -0.21666        -0.035429               39        1

[32561 rows x 15 columns]
```

[20]:
```
y = df.iloc[:,-1]
```

[21]:
```
X = df.iloc[:,:-1]
```

[22]:
```
from sklearn.linear_model import LogisticRegression
```

[23]:
```
logit = LogisticRegression()
```

[24]:
```
from sklearn.model_selection import train_test_split
```

[25]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪25,random_state = 7)
```

[26]:
```
logit.fit(X_train,y_train)
```

```
/home/kirankumar/.local/lib/python3.8/site-
```

```
packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[26]:
```
LogisticRegression()
```

[27]:
```
y_pred = logit.predict(X_test)
```

[28]:
```
from sklearn.metrics import accuracy_score
```

[29]:
```
print("Accuracy of Logistic Regression with scaling :
 ↪",accuracy_score(y_test,y_pred))
```

```
Accuracy of Logistic Regression with scaling : 0.8239773983540105
```

[30]:
```
from sklearn.tree import DecisionTreeClassifier
```

[31]:
```
Dt = DecisionTreeClassifier()
```

[32]:
```
Dt.fit(X_train,y_train)
```

[32]:
```
DecisionTreeClassifier()
```

[33]:
```
y_pred_dt = Dt.predict(X_test)
```

[34]:
```
print("Accuracy of Decision tree with scaling :
 ↪",accuracy_score(y_test,y_pred_dt))
```

```
Accuracy of Decision tree with scaling : 0.8086230192851002
```

[35]:
```
from sklearn.ensemble import RandomForestClassifier
```

[36]:
```
clf = RandomForestClassifier(random_state=0)
clf.fit(X_train, y_train)
```

[36]:
```
RandomForestClassifier(random_state=0)
```

[37]:
```
y_pred_clf = clf.predict(X_test)
```

[38]:
```
print("Accuracy of Random Forest with scaling :
 ↪",accuracy_score(y_test,y_pred_clf))
```

```
Accuracy of Random Forest with scaling : 0.8553003316545879
```