**Project data analysis**

**(Income Data Set)**

**Understanding the Data Science Lifecycle**

**Students Names:**

Hmad Al Marshed & Hussain Al Mansour

**Dr. Adel AL Suliman**

**Professional Master of Data Science**

**Najran University**

**Understanding the Data Science Lifecycle**

In this science project, we used the data science lifecycle that indicates seven steps taken to build, deliver and maintain the "test" data for the income product through extraction, preparation, cleansing, modelling, and evaluation for several independent variables, etc. (*see figure1*)



**Step 1: Business understand**

Before we make collecting the data, we need to understand the objectives in the project. To reduce the put the suitable variables, and reduce time of collecting and understanding of data.

The income data set is an individual’s annual income results from various factors. The train dataset provided predictive feature like education num, occupation, work class, marital status and so on, to predict whether or not these factors can influence the income of the employees.

We explored the possibility in predicting income level (more than or less than 50 K) based on the individual’s personal information through practicing machine learning problem like classification and regression (Customers income prediction)

This is a supervised machine learning problem because here we have a labeled data set.

The dependent variable (Y) is the income.

- The income is divided into two classes: <=50K and >50K

The independent Variables (X) are: Age, education type, occupation and so on.

Number of attributes: 14

- These are the demographics and other features to describe a person

Respiratory data used is: Income datasets (train set) from:

<https://www.kaggle.com/datasets/mastmustu/income>

**Step 2: Data Mining**

We collect data based on variables (age, gender, type of job, income, education level, hours per week, race)

**Open file (lab work)**

In [1]

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file

In [2]

*# Import data*

data = pd.read\_csv("../input/income/train.csv")

data.shape

Out [2]

(43957,15)

In [3]

data.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',

' income'],

dtype='object')

**Step 3: Data Cleaning**

The data I chose is the income dataset for binary classification (https://www.kaggle.com/mastmustu/income). This data set offers a vast array of features. Some columns have a missing value as ? we need to convert actual None.

In[4]

data = data.replace('?', np.nan)

*# Chechking null values*

def about\_data(df):

total\_missing\_values = df.isnull().sum().reset\_index()

total\_missing\_values = total\_missing\_values.rename(columns={'index':'columns',0:'total missing'})

total\_missing\_values['ration of missing'] = total\_missing\_values['total missing']/len(df)

return total\_missing\_values

We find out, there are 3 columns have null value. we can drop it because of percentage of missing value very low.

In[5]

about\_data(data)

Out[5]

|  |  |  |  |
| --- | --- | --- | --- |
|  | columns | total missing | ration of missing |
| **0** | age | 0 | 0.000000 |
| **1** | workclass | 1836 | 0.056386 |
| **2** | fnlwgt | 0 | 0.000000 |
| **3** | education | 0 | 0.000000 |
| **4** | education-num | 0 | 0.000000 |
| **5** | marital-status | 0 | 0.000000 |
| **6** | occupation | 1843 | 0.056601 |
| **7** | relationship | 0 | 0.000000 |
| **8** | race | 0 | 0.000000 |
| **9** | sex | 0 | 0.000000 |
| **10** | capital-gain | 0 | 0.000000 |
| **11** | capital-loss | 0 | 0.000000 |
| **12** | hours-per-week | 0 | 0.000000 |
| **13** | native-country | 583 | 0.017905 |

As we see we found some null value . and we will write the below code to delete null value from our data:

In[6]

data.dropna(inplace=True,axis=0)

about\_data(data)

Out [6]

|  |  |  |  |
| --- | --- | --- | --- |
|  | columns | total missing | ration of missing |
| **0** | age | 0 | 0.000000 |
| **1** | workclass | 0 | 0.000000 |
| **2** | fnlwgt | 0 | 0.000000 |
| **3** | education | 0 | 0.000000 |
| **4** | education-num | 0 | 0.000000 |
| **5** | marital-status | 0 | 0.000000 |
| **6** | occupation | 0 | 0.000000 |
| **7** | relationship | 0 | 0.000000 |
| **8** | race | 0 | 0.000000 |
| **9** | sex | 0 | 0.000000 |
| **10** | capital-gain | 0 | 0.000000 |
| **11** | capital-loss | 0 | 0.000000 |
| **12** | hours-per-week | 0 | 0.000000 |
| **13** | native-country | 0 | 0.000000 |

as we see we have a clean data

**Step 4: Exploratory Analysis**

For this part, we can observe relationships with the data while having fun with functions. In this example, the dataset provided predictive feature like age, education num, employment status, marital status to predict if the income is greater than $50K. It can be used to practice machine learning problem like binary classification. So, we need to accurately predict whether or not someone is making more or less than $50,000 a year.,

Also before start we will explain the name for our columns :

data.columns

In [7]

Index(['age', 'workclass', 'fnlwgt', 'education', 'educational-num',

'marital-status', 'occupation', 'relationship', 'race', 'gender',

'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',

'income\_>50K'],

dtype='object')

Out [7]

|  |  |
| --- | --- |
| Definition | Columns |
| Age of Persons | age |
| Describe work type | workclass |
| Financial Weight | fnlwgt |
| Person's education level | education |
| Years of Experience | educational-num |
| Person's martial status | martial status |
| Person's usual or principal work or business | occupation |
| Gender of Person | gender |
| Person's race | race |
| Person's capital gain | capital gain |
| Person's capital loss | capital loss |
| Earn per hour | hours per hour |
| Persons native country | native country |
| Whether <50k or not | income\_>50K' |

In [8]

data.info()

Out [8]

<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

In[9]

data.describe()

Out[9]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **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 |

we will show some diagram from our database :

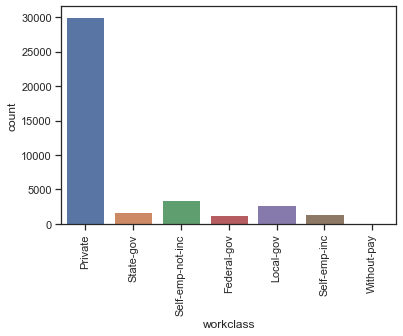
data["workclass"] = [cols.replace(' ', '') for cols in data["workclass"]]

sns.countplot(data=data,x='workclass')

plt.xticks(rotation=90)

In[10]:

Out [10]:



As we see this diagram show for us how many people in workclass and we see the big workclass is a private .

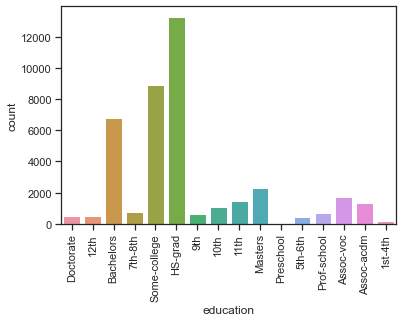
In[11]

data["education"] = [cols.replace(' ', '') for cols in data["education"]]

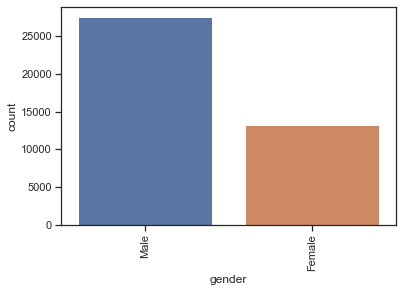
sns.countplot(data=data,x='education')

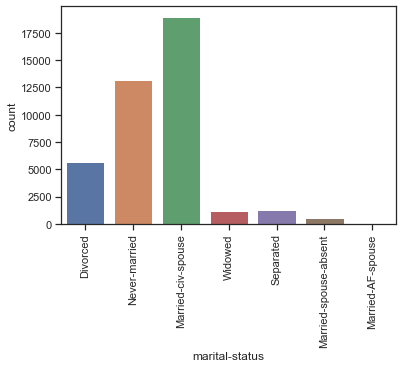
plt.xticks(rotation=90)

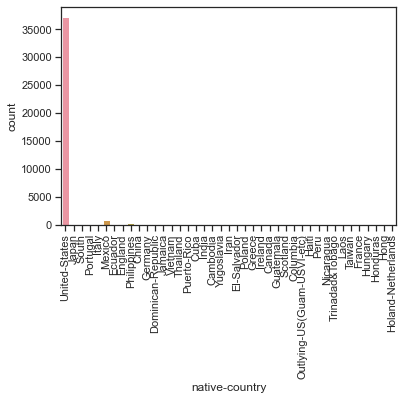
Out [11] :

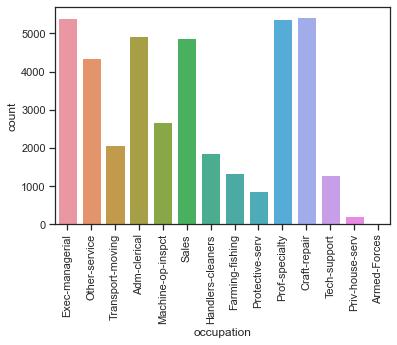


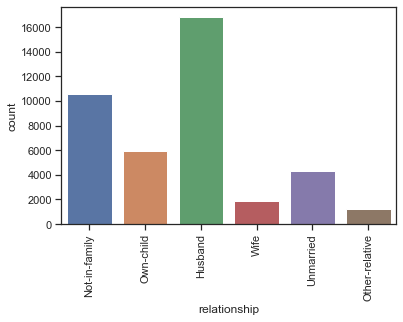
And we will show for all the other diagram but we don't write the code because it's same the last code:











**Step5: Feature Engineering**

In machine learning, a feature is a measurable property or attribute of a phenomenon being observed. If we were predicting the income of customers, a possible feature is the amount of income they get.

We typically perform two types of tasks in feature engineering - feature selection and construction. in this example, I select the top five feature selection.

In [12]

coefs = pd.Series(index=X.columns,data=tuned\_model\_rf.feature\_importances\_[0])

coefs = coefs.sort\_values(ascending=False)[:5]

plt.figure(figsize=(15,10))

sns.barplot(x=coefs.index,y=coefs.values)

plt.xticks(rotation=90)

Out [12]

(array([0, 1, 2, 3, 4]),

[Text(0, 0, 'age'),

Text(1, 0, 'fnlwgt'),

Text(2, 0, 'native-country\_Haiti'),

Text(3, 0, 'native-country\_Guatemala'),

Text(4, 0, 'native-country\_Greece')])

****

**Step 6: Predictive Modeling**

I used the predictive modeling to ensure that the outcomes from the model actually make sense and are significant. Then, it is critical that you evaluate its success. A process called k-fold cross validation. This allows the model to be trained on all the data instead of using a typical train-test split.

In this example, I used (logisticRegression(LR), KNeighborsClassifier(KNN), DecisionTreeClassifier(DTC), RandomForestClassifier(RFC)).

In [13]

trees = 100

max\_features = 3

results = []

names\_of\_models = []

model\_list =[('LR', LogisticRegression()),

('KNN', KNeighborsClassifier()),

('DTC', DecisionTreeClassifier()),

('RFC', RandomForestClassifier(n\_estimators=trees,max\_features=3))]

for name, model in model\_list:

kfold = KFold(n\_splits=10, random\_state=None)

cv\_results = cross\_val\_score(model, scaled\_X\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

names\_of\_models.append(name)

res = "{}: {} ({})".format(name, cv\_results.mean(), cv\_results.std())

print(res)

**Out [13]**

LR: 0.8195238593199343 (0.006410311684477739)

KNN: 0.8247283318387449 (0.004367257918147866)

DTC: 0.9133345796457777 (0.0037535594855644784)

RFC: 0.9258656297610204 (0.0032177164323382765)

As indicated above, Random Forest has highest accuracy. Then, we choose random forest model and tune params

**Step7: Data visualization**

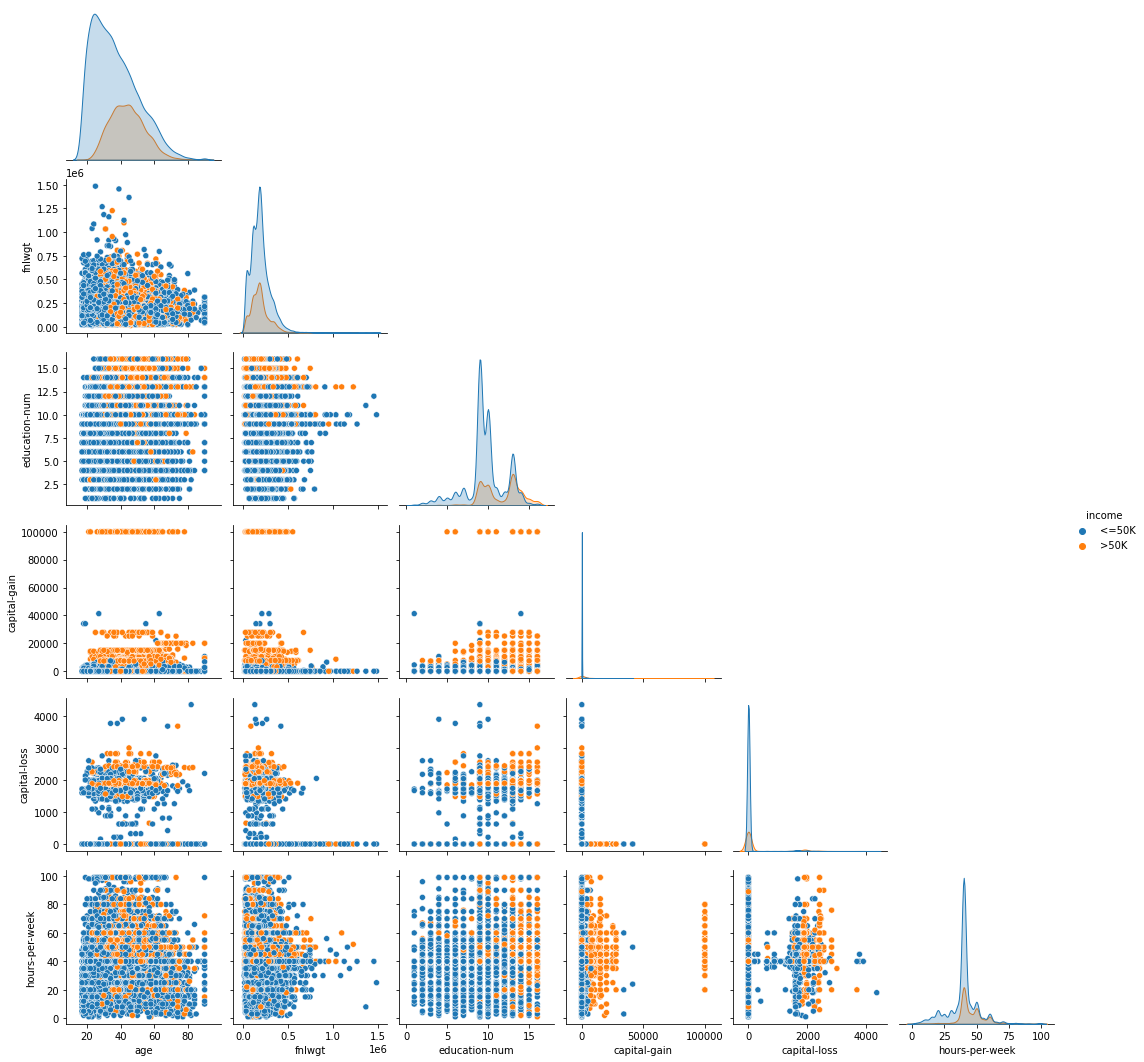
Here we represent the model in way that the different stakeholders in the project can understand.

In [13]

sns.pairplot(data,hue='income\_>50K',corner=True)

Out[14]

<seaborn.axisgrid.PairGrid at 0x1e0e4114be0>



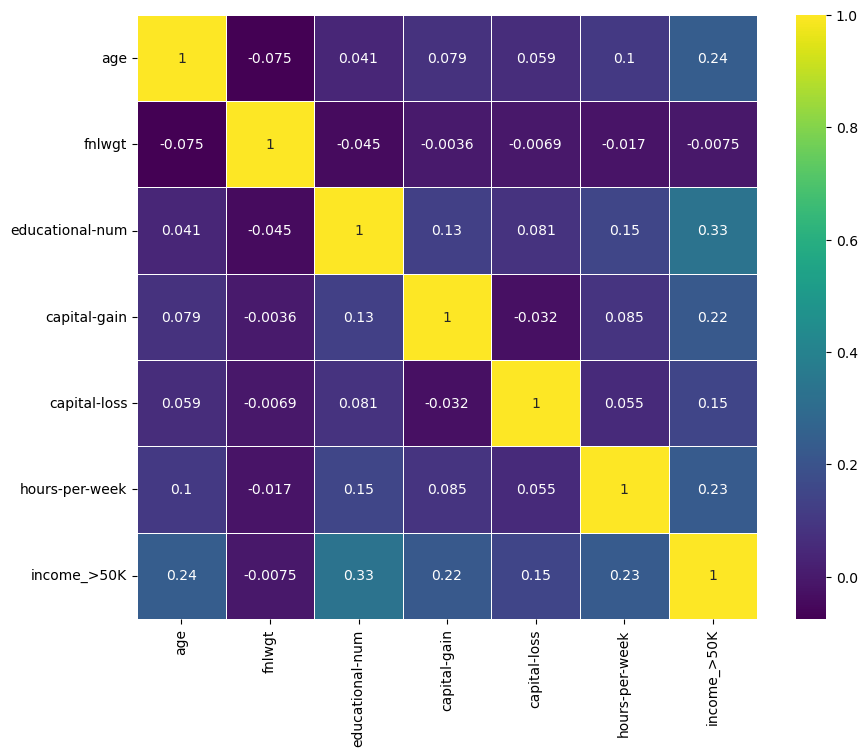
In [15]

plt.figure(figsize=(10,8),dpi=100)

sns.heatmap(data.corr(),cmap="viridis",annot=True,linewidth=0.5)

<AxesSubplot:>

out [15]



In one hand, the positive relationship with the income are:Age, Workclass Federal-, Workclass private, fnlwgt, ..all the variables in blue color.

While the negative relationship with the income are: Workclass never worked, Education 7th – 8 th, Martial status never married, Occupation family finshing …. All the variables in green colors.

On other hand, there are some variables such as: educationPreschool , relationshipUnmarried , and occupationTransport-moving… all the variables in black color. These are not influence the income at all.

**Conclusion**

There are little number of employees make more than 50 K. We recommended to increase the hours per week to 40., also we can employee the worker with bachelors and master degree (need to increase) for income less than 50 K. Prefer work in private class for high income. In sum, employees should encourage to increase their income with the variables that are related to increase the income such as age and avoid all other variables that are not impact the income or has a negative impact on it.