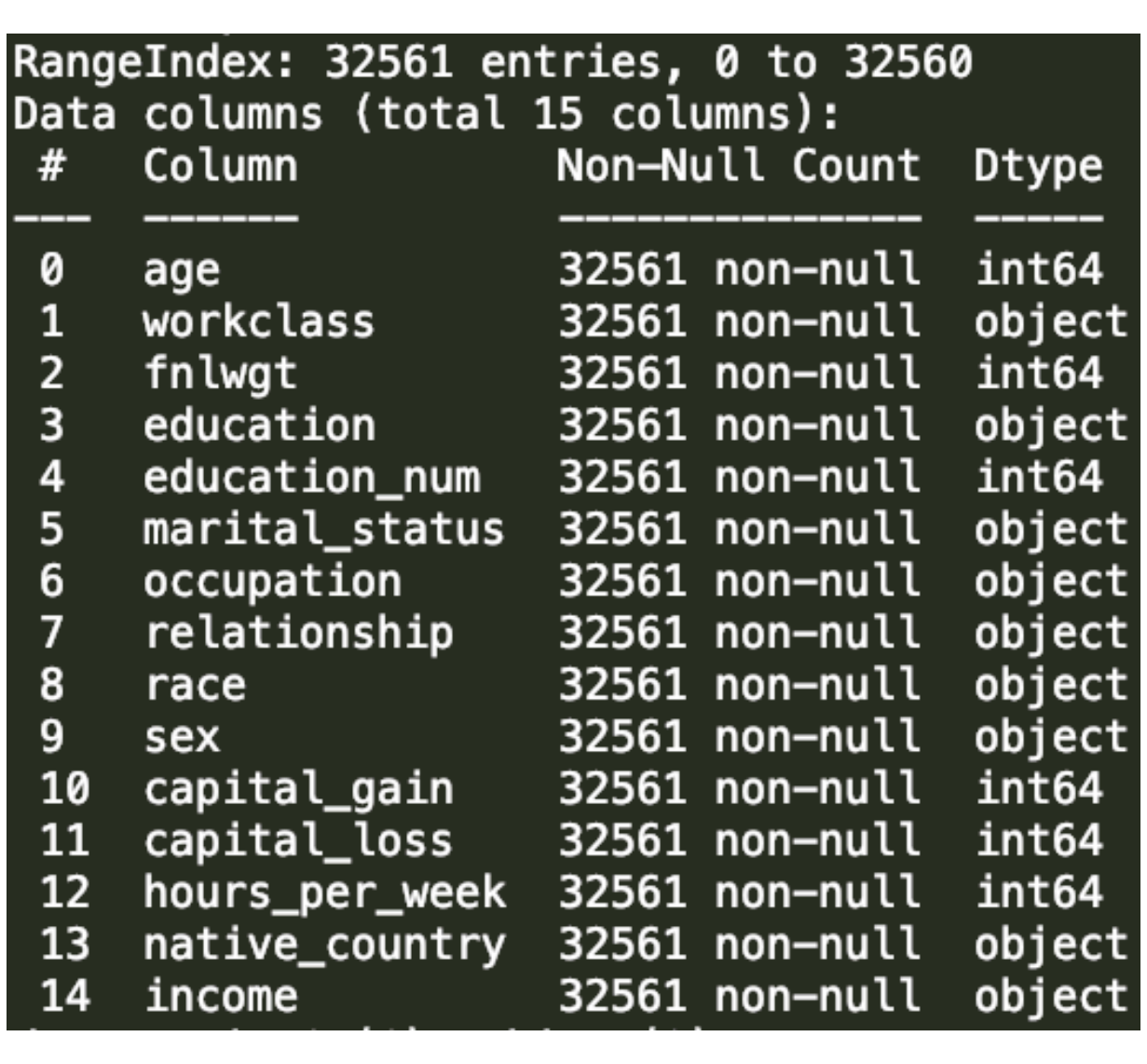
HW2: Cross Validation

309554001 劉雨恩

1. Data Evaluation

1.1 Whole Data



```
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
```

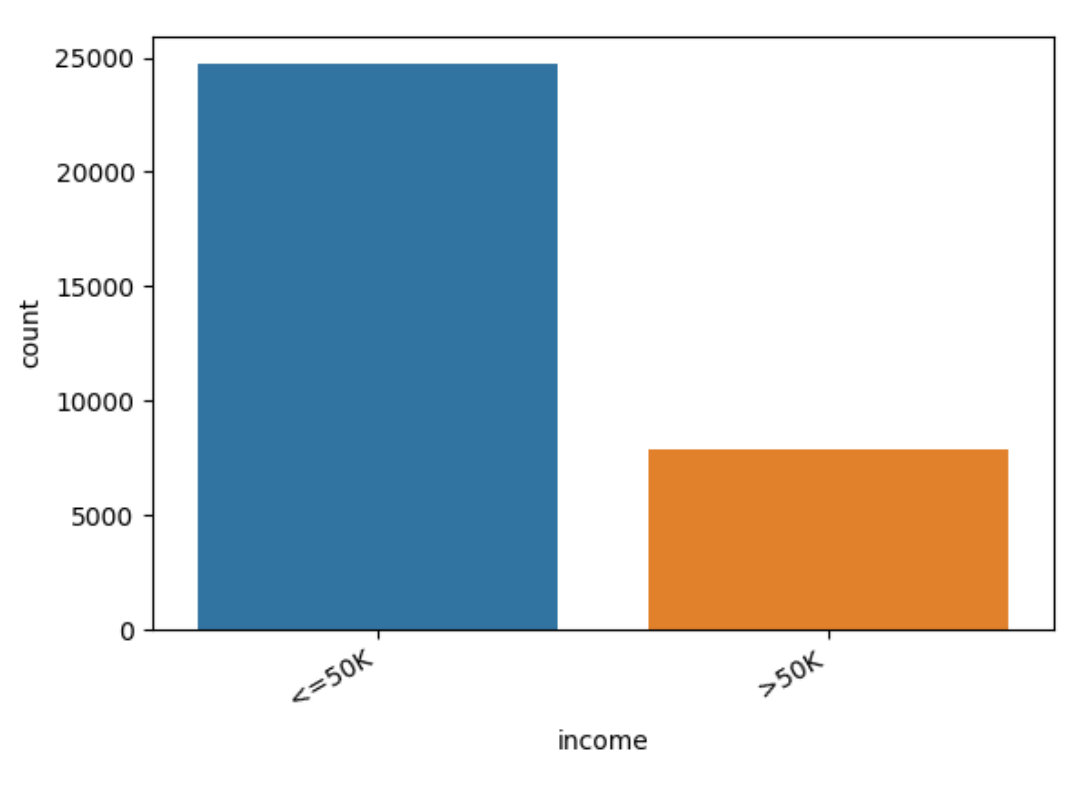
- 首先使用 info() 來檢視資料集，本資料集裡有 32561 筆資料，且每個類別都無缺失資料

```
# read data
data = pd.read_csv('HW2data.csv')
data.info()
```

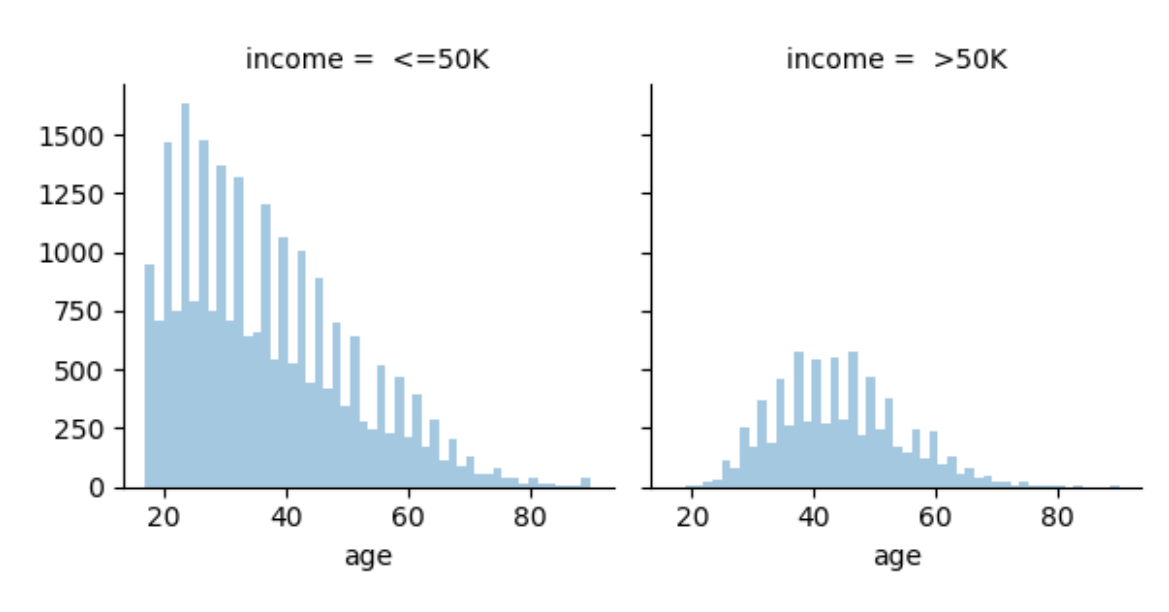
1.2 Pair Comparing

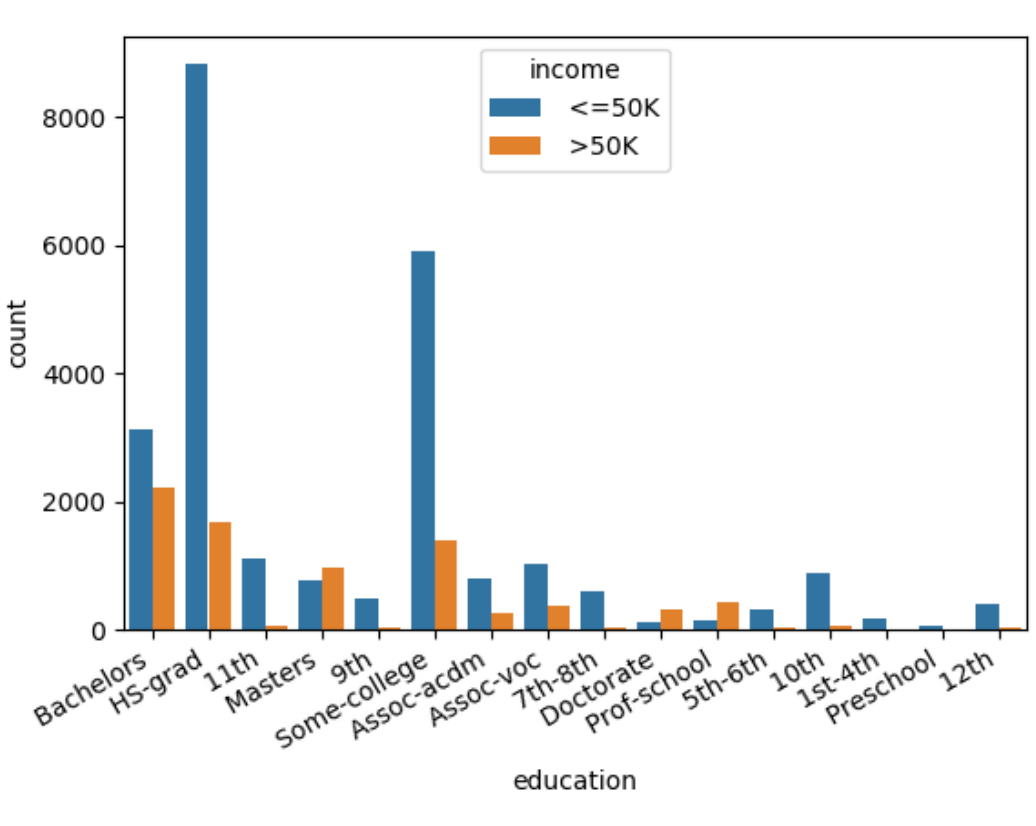
檢視各個 label 與 `income` 的關係

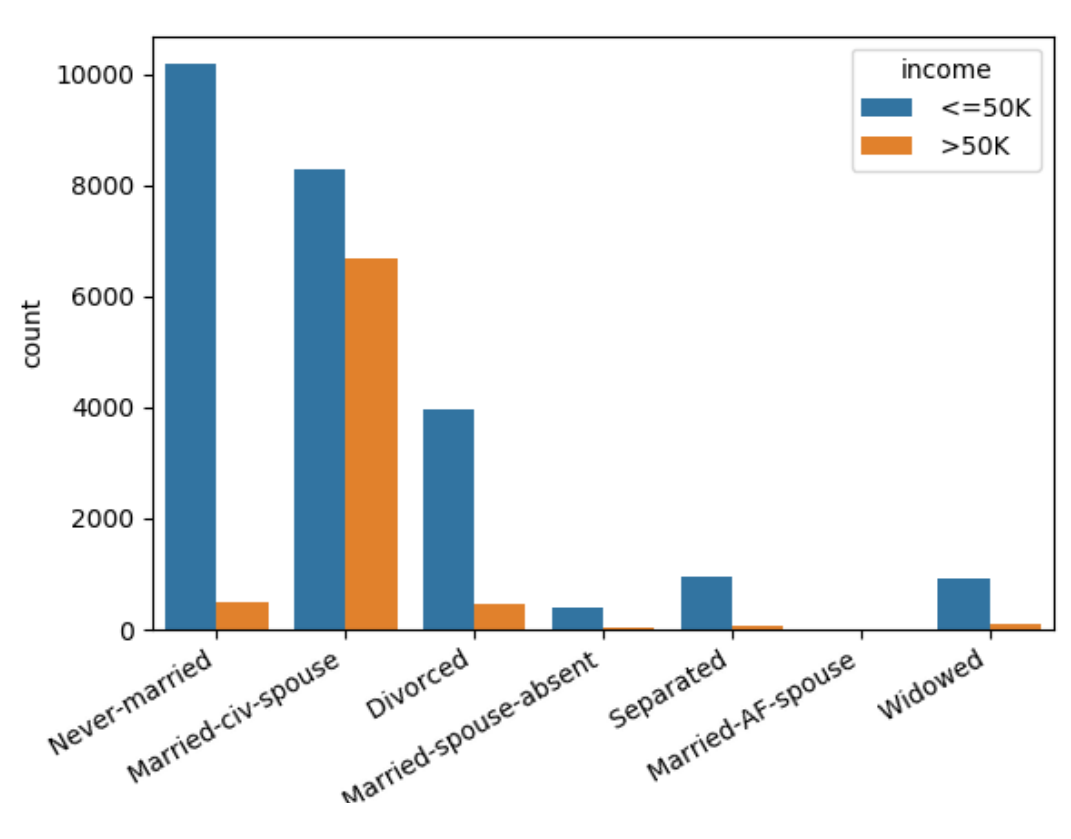
- `income` : <= 50K 的大概為 7.5 成，而 > 50K 大概為 2.5 成

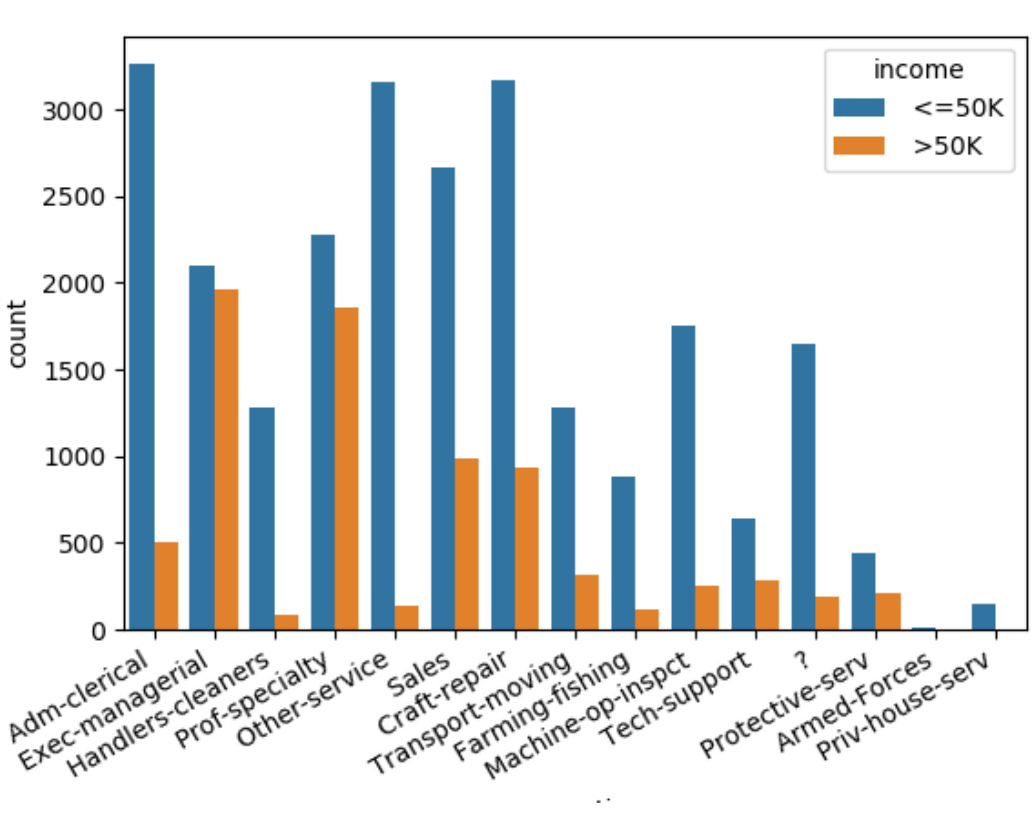


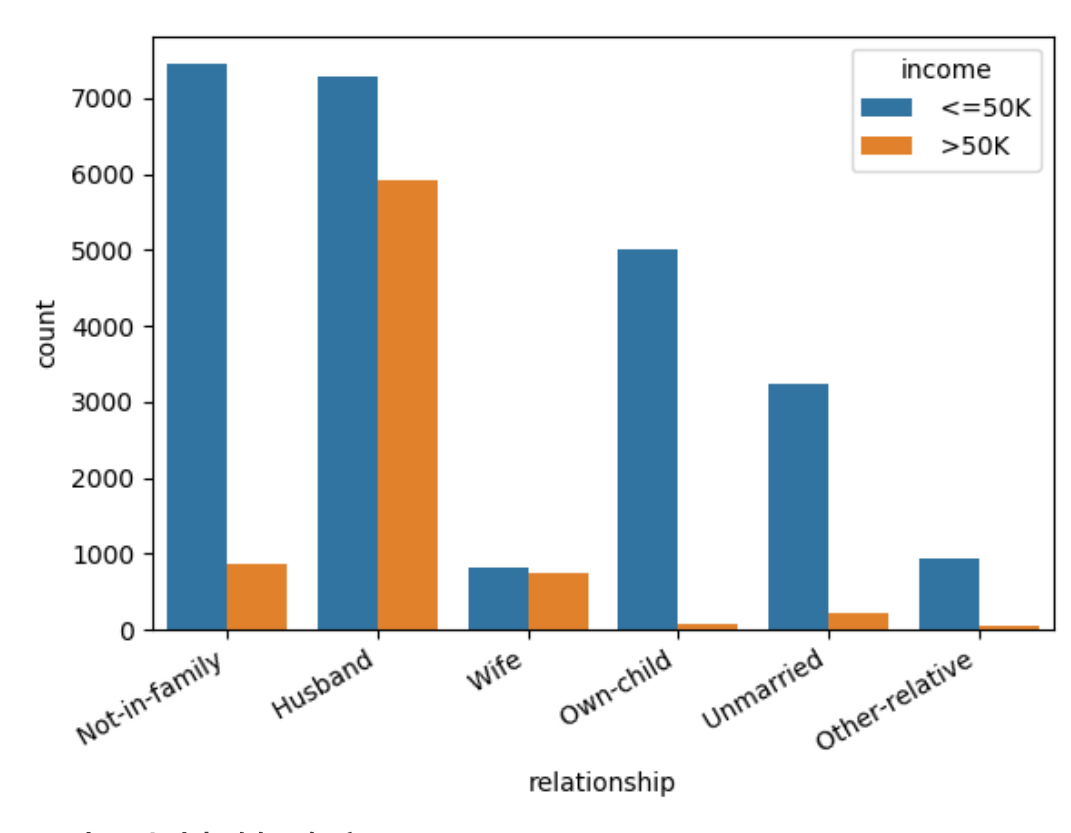
- `age` : <= 50K 大都分佈於 20~40 歲，> 50K 大都分佈於 40、50 歲上下
- `education` : 除了 `Bachelors` , `Masters` , `Doctorate` 和 `Prof-school` > 50K 和 <= 50K 的比例差不多或超過，其他 > 50K 都明顯比 <= 50K 的比例低很多
- `marital_status` : 除了 `Married-civ-spouse` > 50K 和 <= 50K 的比例差不多，其他 > 50K 都明顯比 <= 50K 的比例低很多
- `occupation` : 除了 `Exec-managerial` 和 `Prof-specialty` > 50K 和 <= 50K 的比例差不多，其他 > 50K 都明顯比 <= 50K 的比例低很多
- `relationship` : 除了 `Husband` 和 `Wife` > 50K 和 <= 50K 的比例差不多，其他 > 50K 都明顯比 <= 50K 的比例低很多
- `sex` : 依比例來看，女性 <= 50K 比男性高很多

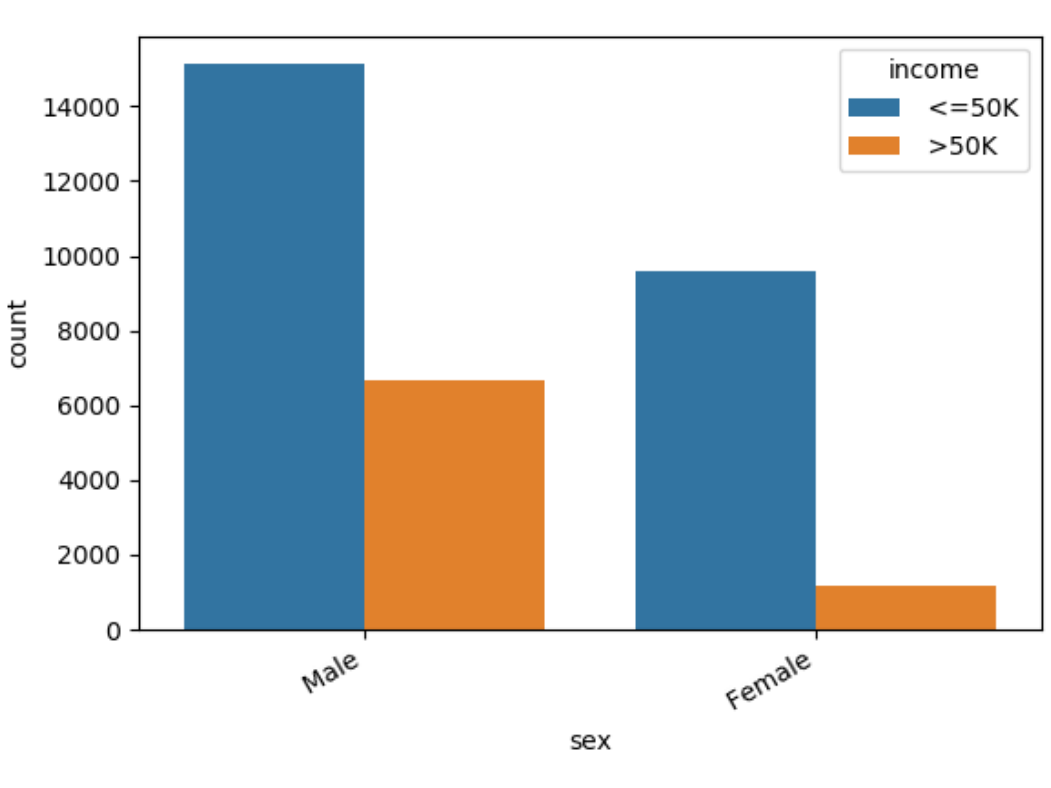




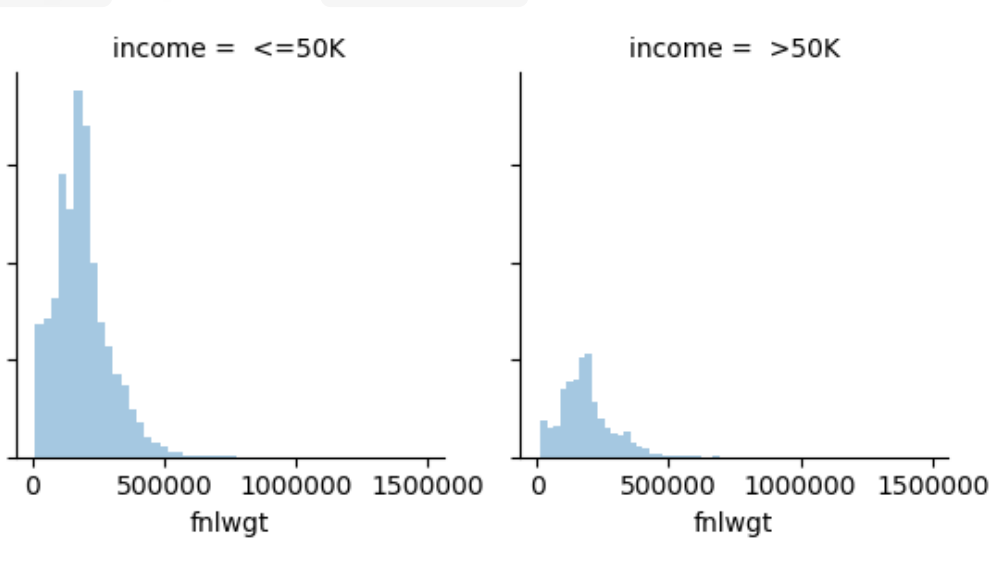




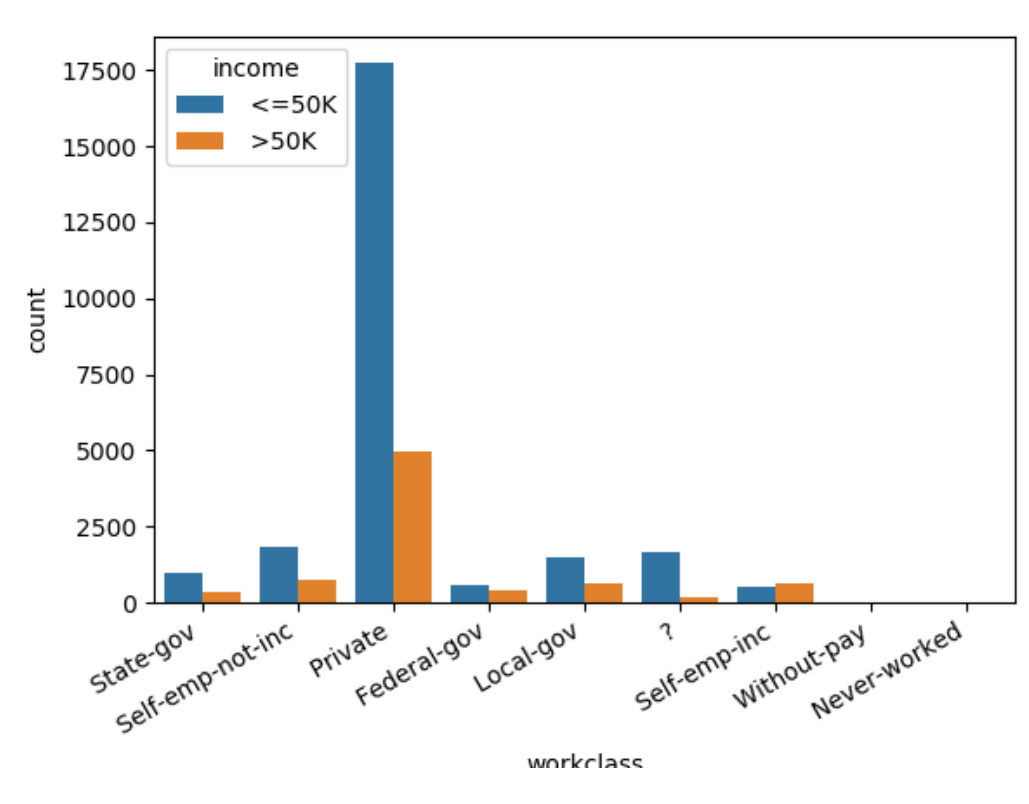


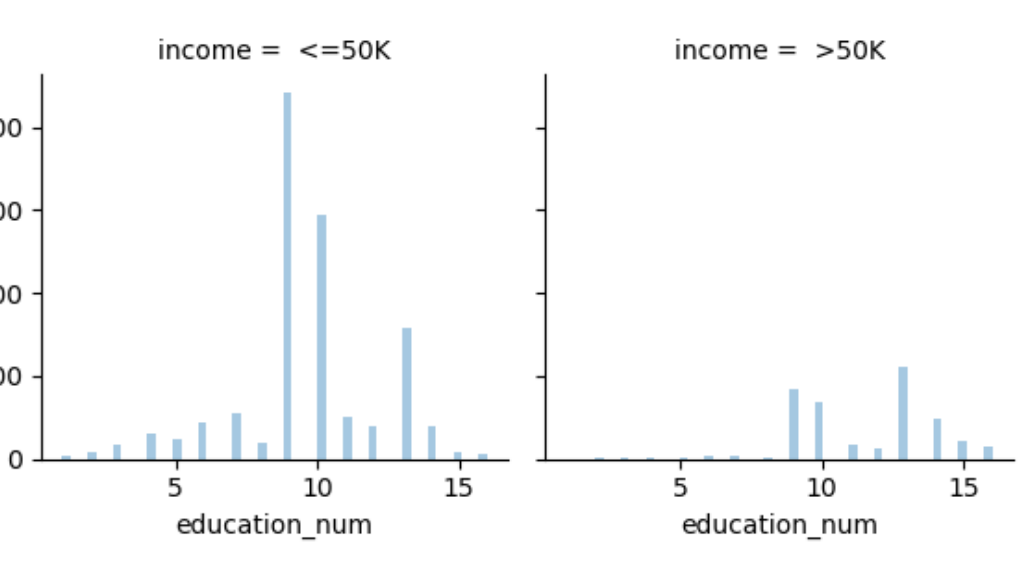


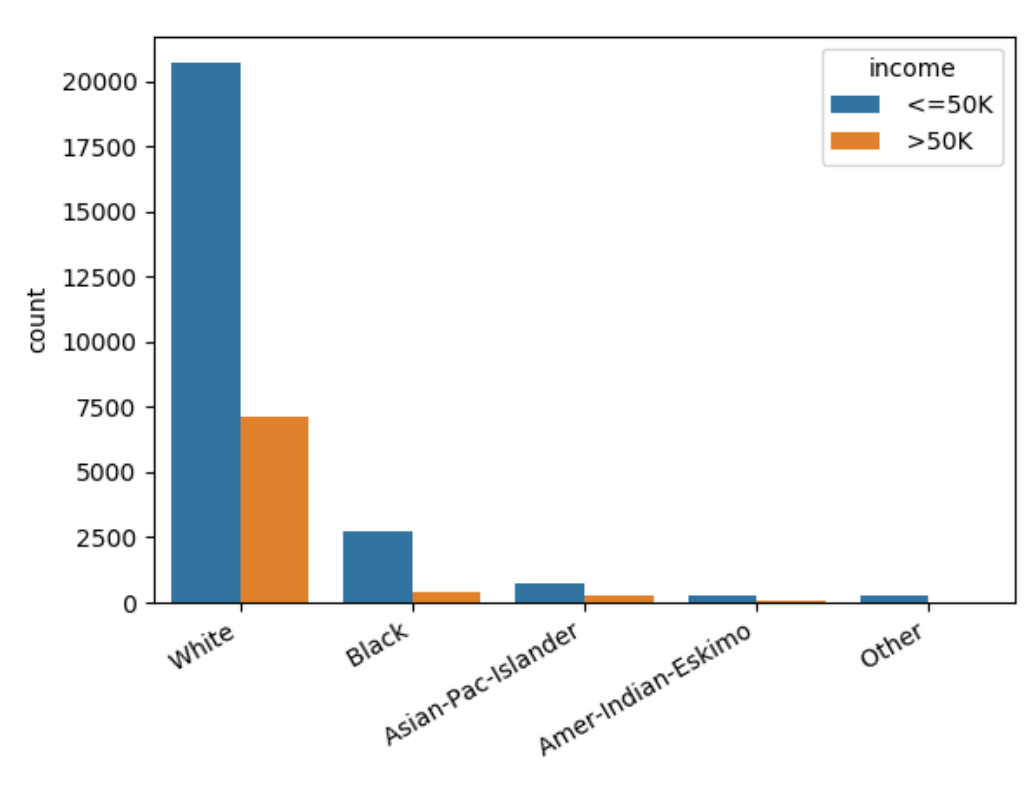
- 需去除掉的資訊
  - `fnlwgt` : 序號和 `income` 無關係

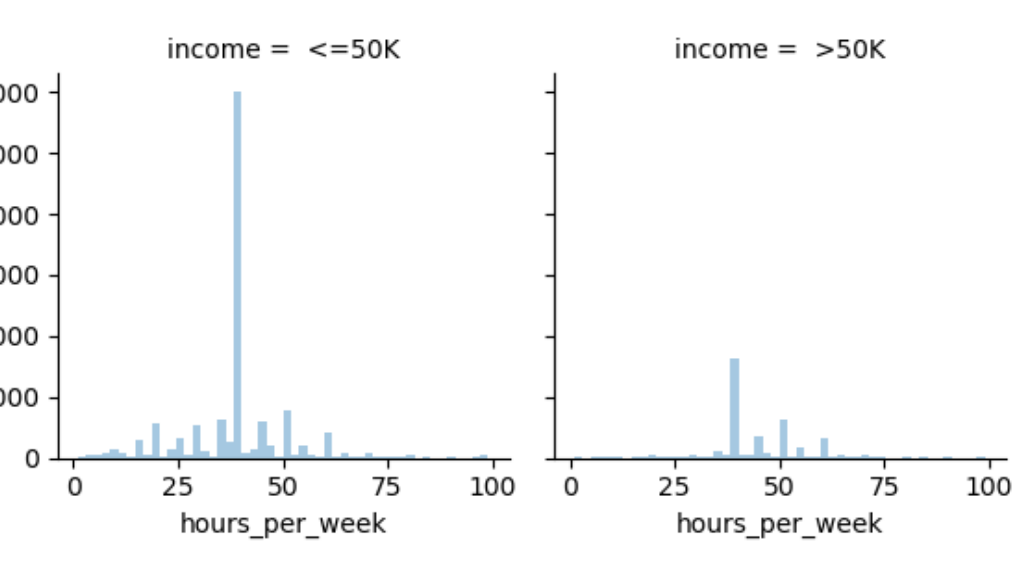


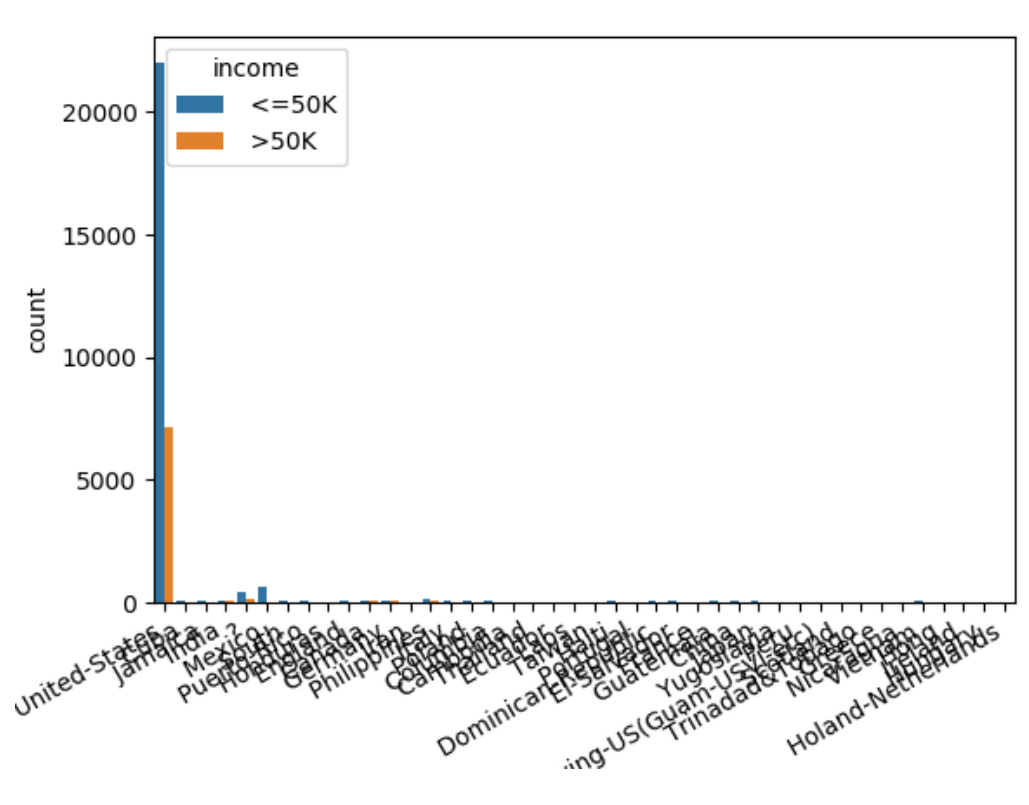
  - `workclass` , `education_num` , `race` , `hours_per_week` , `native_country` , 和 `capital_total` (`capital_gain` - `capital_loss`): 這些類別的資料量資料分佈極不均 (在某個區間偏多，其他區間明顯不足)

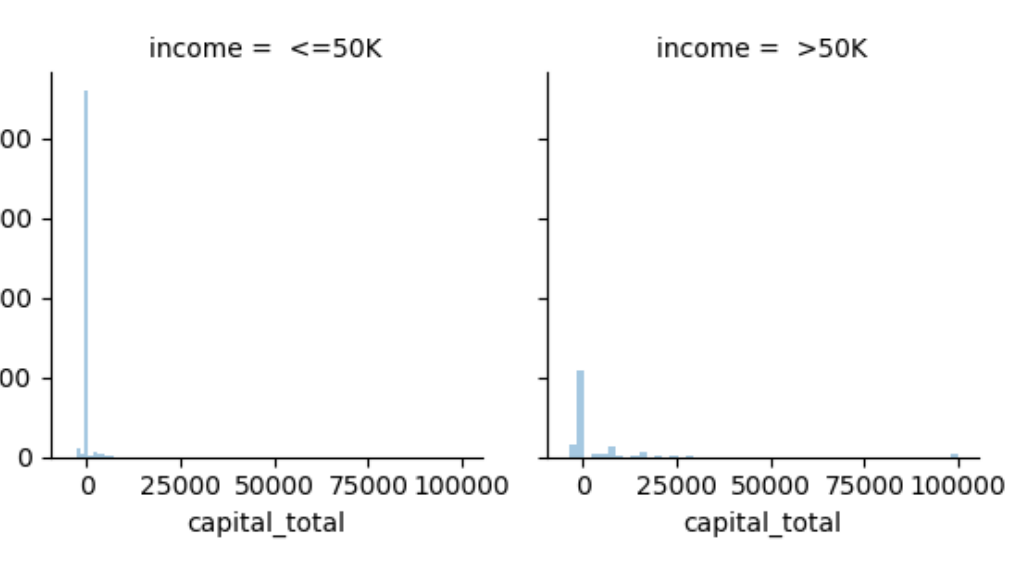













```python
def plot_pair_count(data, x_label, y_label='income', \
                    pics_path = './pics/', i=0):
    plt.clf()

    fig = plt.figure()
    if x_label == y_label:
        sns.countplot(data[x_label])
    elif x_label == 'age' or x_label == 'fnlwgt' \
        or x_label == 'education_num' or x_label == 'capital_total' \
        or x_label == 'hours_per_week':
        g = sns.FacetGrid(data, col=y_label)
        g.map(sns.distplot, x_label, kde=False)
    else:
        sns.countplot(data[x_label], hue=data[y_label])

    fig.autofmt_xdate()
    plt.savefig(pics_path + str(i) + '_' + x_label + '.png')

# data evaluation
data['capital_total'] = data['capital_gain'] - data['capital_loss']
data = data.drop(columns=['capital_gain', 'capital_loss'])
for i, label in enumerate(data.columns):
    print(label)
    plot_pair_count(data, label, i=i)
```

2. Data Preprocessing

2.1 Drop columns

去除不需要的資訊，以減輕模型負擔

```python
x_data = data.drop(columns=['fnlwgt', 'income', 'workclass', \
                            'education_num', 'race', 'hours_per_week', \
                            'native_country', 'capital_total'])
y_data = data['income']
```

2.2 Label Encoding

將 data 裡的 `education` , `marital_status` , `occupation` , `sex` 和 `income` encode 轉為數字

```python
label_encoder = preprocessing.LabelEncoder()
x_data['education'] = label_encoder.fit_transform(x_data['education'])
x_data['marital_status'] = label_encoder.fit_transform(x_data['marital_status'
x_data['occupation'] = label_encoder.fit_transform(x_data['occupation'])
x_data['relationship'] = label_encoder.fit_transform(x_data['relationship'])
x_data['sex'] = label_encoder.fit_transform(x_data['sex'])
y_data = label_encoder.fit_transform(y_data)
```

3. 10-fold Cross Validation

使用 Random Forest Classifier 做為 model

```python
def K_fold_CV(k, x, y):
    sub_size = len(y)//k
    last_remain = len(y)%k

    train_acc, test_acc = 0, 0
    for i in range(k):
        if i < (k-1):
            idx_seq = [j for j in range(sub_size*i, sub_size*(i+1))]
        else:
            idx_seq = [j for j in \
                        range(sub_size*i, sub_size*(i+1)+last_remain)]

        test_x = x[idx_seq[0]:idx_seq[-1]+1]
        test_y = y[idx_seq[0]:idx_seq[-1]+1]
        train_x = x.drop(idx_seq, axis=0)
        train_y = np.delete(y, idx_seq, axis=0)

        # random forest model
        forest = RandomForestClassifier(oob_score=True)
        forest.fit(train_x, train_y)
        train_acc += forest.oob_score_

        # predict
        pred_y = forest.predict(test_x)
        test_acc += accuracy_score(test_y, pred_y)

    return train_acc / k, test_acc / k

train_acc, test_acc = K_fold_CV(10, x_data, y_data)
print(train_acc, test_acc)
```

4. Results

- Average training accuracy: `0.8081651971184474`
- Average testing accuracy: `0.8082371629731163`