Data Science - Homework 3

- Motivation

I will use adult dataset from UCI as the dataset for this assignment, it also known as "Census Income" dataset. After importing the dataset, I found that it has 32561 records and 15 attributes in training set, and then I check its details, as can be seen that there are missing values in three columns, including workclass, occupation and native_country.

```
df_adult_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age              32561 non-null int64
workclass        30725 non-null object
fnlwgt           32561 non-null int64
education        32561 non-null object
education_num    32561 non-null int64
marital_status   32561 non-null object
occupation       30718 non-null object
relationship     32561 non-null object
race             32561 non-null object
gender           32561 non-null object
capital_gain     32561 non-null int64
capital_loss     32561 non-null int64
hours_per_week   32561 non-null int64
native_country   31978 non-null object
income_bracket   32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

- Use Python to analyze the dataset with missing data (NA) that I select. (IDE: jupyter)

1. Replacing " ? " by NaN to facilitate the analysis in subsequent steps

```
df_adult_train.replace("?", np.nan, inplace = True)
df_adult_train.head()
```

2. Checking for missing values

The symbol is now replaced with NaN which is null value, these are easily recognized and computed to check for the summation of the missing data available in the dataset.

From this statistics I get a total number of 1836 workclass values, 1843 occupation values, and 583 native country values missing from the existing dataset. All the fields of containing missing values are categorical data.

```
missing_data = df_adult_train.isnull()
missing_data.sum()
```

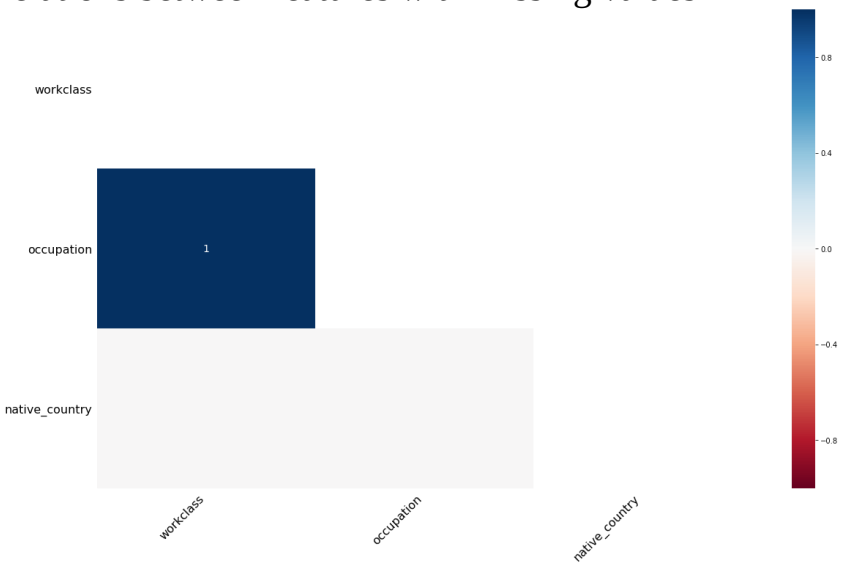
```
age                 0
workclass        1836
fnlwgt              0
education           0
education_num       0
marital_status      0
occupation       1843
relationship        0
race                0
gender              0
capital_gain        0
capital_loss        0
hours_per_week      0
native_country    583
income_bracket      0
dtype: int64
```

```
Column:  workclass
Missing Data: 1836 (5.64%)
Data Type:  object

Column:  occupation
Missing Data: 1843 (5.66%)
Data Type:  object

Column:  native_country
Missing Data: 583 (1.79%)
Data Type:  object
```

3. Show correlations between features with missing values



The heatmap is used to represent the correlation between features with missing values, that is, the degree of correlation between missing values for one feature when another feature has missing values. If there are no numbers in the heatmap , indicating that there is no missing correlation between features.

From the heatmap above, I could observed the correlation between workclass and occupation is 1, it means that when workclass is missing, occupation must also be missing, this two columns could be Missing Not at Random (MNAR), so I would have no information in my dataset to impute these missing values. Generally, we assume Missing At Random (MAR) whenever possible just to avoid this situation.

- Try to solve the issues of missing values or another data errors.

Because missing values in this dataset are categorical data, and I assumed types of missing values are MAR, I tried three methods to handle this problem. One of the methods I used is replacing missing values with SimpleImputer() which scikit-learn provides. And why choose SimpleImputer()? I think that SimpleImputer() is best for cases where there are categorical data which need to handle with imputation, and missingness in one value is not affected by other values. My wild guess is that KNNImputer() would be good for imputing numeric data. The second method is imputation with Multiple Imputation by Chained Equations (MICE). Finally, I would try to create a new category for missing values.

- Replacing missing values with the most frequent values

```
imr = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imr = imr.fit(df_adult_train.values)
imputed_data = imr.transform(df_adult_train.values)
imputed_data
```

```
array([[39, 'State-gov', 77516, ..., 40, 'United-States', 0],
       [50, 'Self-emp-not-inc', 83311, ..., 13, 'United-States', 0],
       [38, 'Private', 215646, ..., 40, 'United-States', 0],
       ...,
       [58, 'Private', 151910, ..., 40, 'United-States', 0],
       [22, 'Private', 201490, ..., 20, 'United-States', 0],
       [52, 'Self-emp-inc', 287927, ..., 40, 'United-States', 1]],
      dtype=object)
```

- Imputation with MICE

```
#create mice imputer
mice_imputer = IterativeImputer()

#fit and transform on train and transform test
mice_train.iloc[:, :] = (mice_imputer.fit_transform(mice_train))
```

- Creating a new category for missing values

1. Drop the meaningless columns (mentioned in the previous assignment)

```
df_adult_train = df_adult_train.drop(["fnlwgt","education","capital_gain","capital_loss"], axis = 1)
```

First, I ran a for loop over all the columns which gets the count of unique values. I can observe that some columns have a lot of distinct values like fnlgwt attribute which has around 20000+ values, it may be noisy data for my model.

Second, the two columns education and education_num have the same meaning, so I removed one of them.

Third, I found through the histogram that capital_loss and capital_gain have many zero values, so I removed them.

2. Impute null value with new category

```
# Function to impute null value with new category
def impute_nan_create_category(DataFrame,ColName):
     DataFrame[ColName] = np.where(DataFrame[ColName].isnull(),"Unknown",DataFrame[ColName])

for Columns in ['workclass','occupation','native_country']:
    impute_nan_create_category(create_train, Columns)
```

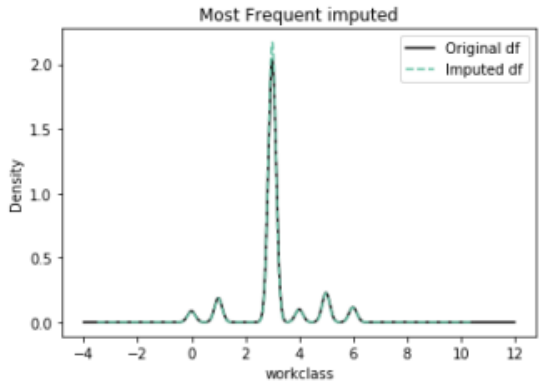
3. Encoding

```
le = LabelEncoder()
for i in colname:
    create_train[i] = le.fit_transform(create_train[i])
    le_name_mapping = list(zip(le.classes_, le.transform(le.classes_)))
    print("Feature :", i)
    print("Mapping :", le_name_mapping)
```

- Explain the results I obtain.

After imputing the missing values with three methods, I found that MICE imputer appears to work better. The following analysis takes workclass as an example.

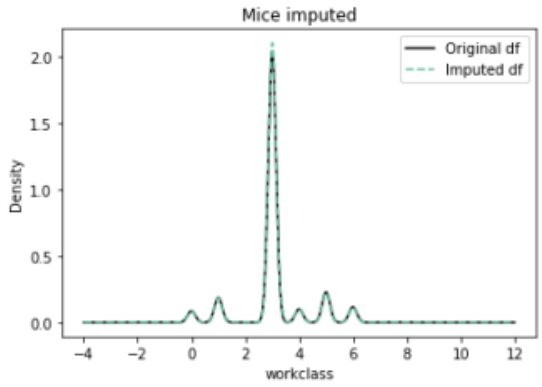
Replacing missing values with the most frequent values




```
#most frequent imputing is as good as the mice inputing
most_frequent_train[1].var(), train['workclass'].var()
```

```
(1.2258784627031534, 1.3025791283407606)
```

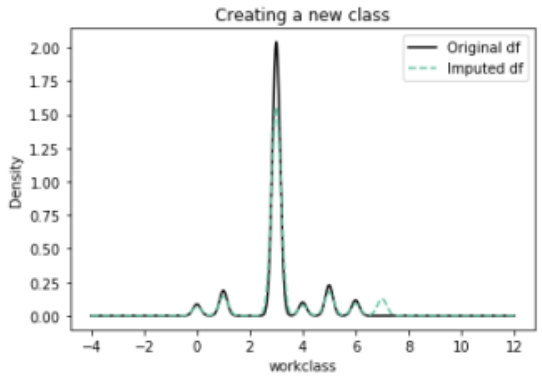
Imputation with MICE




```
#mice imputing barely has an impact on variance
mice_train['workclass'].var(), train['workclass'].var()
```

```
(1.230194446913309, 1.3025791283407606)
```

Creating a new category for missing values




```
#creating a new class incareses the variance in this feature
create_train['workclass'].var(), train['workclass'].var()
```

```
(2.0382100988750618, 1.3025791283407606)
```

MICE imputer preserves the shape of workclass and the amount of variance better than the other two strategies. Creating a new category for missing values has the most dramatic impact on the distribution of the workclass.

On the other hand, I analyzed the advantages and disadvantages of three strategies.

	advantage	disadvantage
Frequent Categorical Imputation	Simple and easy to implement for categorical data.	1. Features having a max number of null values may bias prediction if replace with the most frequent values. 2. It distorts the relation of the most frequent label.
MICE	1. It runs several iterations of regression models on our dataset 2. It can produce great results when data is MAR.	It can be a bit more computationally
Create a New Category	Simple and easy to implement for categorical data.	1. It may create random data if the missing values are more. 2. It doesn't give good results when missing value is a high percentage of the data.

- Discuss possible problems I plan to investigate for future studies

In the previous assignment, I mentioned that this dataset may exist a data imbalance problem. As a first step, I should strive to get more real-world data so that the data represents the real-world problem. If I cannot get additional data on minority classes, then I may resort to implementing techniques that can mitigate the imbalance like undersampling.

Maybe I will show that the dataset can be used to give sufficiently higher accuracy by reducing the size of the training dataset and with the use of any techniques for imbalance. If the result is negative, I can say that no imbalance technique may be required to make good predictions.