Module 1 Project: Understanding Income Inequality

Kejiang Yao

College of Professional Studies, Northeastern University

Professor: Dr. Mary Donhoffner

Date: 09/26/2022

Introduction

For this week's assignment, we are given the census data (*adult-all* dataset) with attributes of US citizens such as occupation, education, gender, race, etc. To assist organizations on working to ensure equal pay, we are planning to build a model that would achieve the goal of accurately classify low income from high income citizens. Meanwhile, this model should also provide us insights about attributes contribute to affluency and how can we improve policies in the US.

There are two parts needed to be complete in this assignment:

Part 1:

Use proper data cleansing techniques to ensure that you have the highest quality data to model this problem. Detail your process and discuss the decisions you made to clean the data.

Part 2:

Build a nearest neighbors model with the given data, interpret the results, and convey those results to stakeholders. Highlight key learning points such as feature importance of variables, how those variables explain the scenario, how you determined K, why you choose that final value of K, and the overall accuracy of your model and accompanying models.

Both of these two parts will be completed using Python on Jupyter Notebook. Necessary codes and visuals will be attached directly within this report, and the full python script will be submitted separately in another file.

Part 1: Data Cleaning

Firstly, since the ***adult-all.csv*** is a local csv file, we need to load this dataset to the environment for further processing. By checking the csv file, I did not find column name existed for each column. Therefore, I added column name for each column according the ***week1-dataset-variabletypes.pdf***. The screenshot below is the code used for loading data, adding name for each Column, and first five rows of the dataset.

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv ('/Users/kejiangyao/Desktop/ALY 6020/Module 1 Project – Understanding Income Inequality/adult-all.csv'
```

```
df.columns = ['Age', 'Workclass', 'fnlwgt', 'Education','Education_num','Marital_status','Occupation','Relationship','R
```

```
df.head()
```

	Age	Workclass	fnlwgt	Education	Education_num	Marital_status	Occupation	Relationship	Race	Sex	Capital_gain	Capital_loss	Hours_per_week	Na
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	U
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	U
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	U
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	U

After loading the dataset into the environment, we could start looking for some basic insights through the summary statistics on each variable. Below is the screenshot of summary statistics of all numeric variables. There are 6 numeric variables and 48841 records in total. For example, the average age of these observations is 38.64, standard deviation is 13.71, minimum age is 17, and maximum age is 90.

	Age	fnlwgt	Education_num	Capital_gain	Capital_loss	Hours_per_week
count	48841.000000	4.884100e+04	48841.000000	48841.000000	48841.000000	48841.000000
mean	38.643578	1.896664e+05	10.078029	1079.045208	87.504105	40.422391
std	13.710650	1.056039e+05	2.570965	7452.093700	403.008483	12.391571
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175550e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781470e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376460e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

There are 9 categorical variables in this dataset, which are Workclass, Education, Marital_status, Occupation, Relationship, Race, Sex, Native_country, and Salary. The screenshot below provides the number of observations for each category within each variable.

	Workclass
Private	33906
Self-emp-not-inc	3862
Local-gov	3136
?	2799
State-gov	1980
Self-emp-inc	1695
Federal-gov	1432
Without-pay	21
Never-worked	10

	Education
HS-grad	15784
Some-college	10878
Bachelors	8024
Masters	2657
Assoc-voc	2061
11th	1812
Assoc-acdm	1601
10th	1389
7th-8th	955
Prof-school	834
9th	756
12th	657
Doctorate	594
5th-6th	509
1st-4th	247
Preschool	83

	Marital_status
Married-civ-spouse	22379
Never-married	16116
Divorced	6633
Separated	1530
Widowed	1518
Married-spouse-absent	628
Married-AF-spouse	37

	Occupation
Prof-specialty	6172
Craft-repair	6112
Exec-managerial	6086
Adm-clerical	5610
Sales	5504
Other-service	4923
Machine-op-inspct	3022
?	2809
Transport-moving	2355
Handlers-cleaners	2072
Farming-fishing	1490
Tech-support	1446
Protective-serv	983
Priv-house-serv	242
Armed-Forces	15

	Relationship
Husband	19716
Not-in-family	12582
Own-child	7581
Unmarried	5125
Wife	2331
Other-relative	1506

	Race
White	41761
Black	4685
Asian-Pac-Islander	1519
Amer-Indian-Eskimo	470
Other	406

	Sex
Male	32649
Female	16192

	Native_country
United-States	43831
Mexico	951
?	857
Philippines	295
Germany	206
Puerto-Rico	184
Canada	182
El-Salvador	155
India	151
Cuba	138
England	127
China	122
South	115
Jamaica	106
Italy	105
Dominican-Republic	103
Japan	92
Guatemala	88
Poland	87
Vietnam	86
Columbia	85
Haiti	75
Portugal	67
Taiwan	65
Iran	59
Greece	49
Nicaragua	49
Peru	46
Ecuador	45
France	38
Ireland	37
Hong	30
Thailand	30
Cambodia	28
Trinadad&Tobago	27
Laos	23
Yugoslavia	23
Outlying-US(Guam-USVI-etc)	23
Scotland	21
Honduras	20
Hungary	19
Holand-Netherlands	1

	Salary
<=50K	37154
>50K	11687

As we can see from each categorical variable, there is a category called ***'?'*** which is the null value. Therefore, I convert this value into NaN and count the number of it within each categorical variable as shown in the screenshot below.

```
df = df.replace('?', np.nan)
```

```
pd.DataFrame(df.isnull().sum())
```

	0
Age	0
Workclass	2799
fnlwgt	0
Education	0
Education_num	0
Marital_status	0
Occupation	2809
Relationship	0
Race	0
Sex	0
Capital_gain	0
Capital_loss	0
Hours_per_week	0
Native_country	857
Salary	0

There are 2799 null values from ***Workclass***, 2809 from ***Occupation***, and 857 from ***Native_country***. If we impute these nulls to the most common value within each variable, it will influence the quality and accuracy of the data. Since the sum of the number of null-value records is 6465 which is around 13% of the dataset. Therefore, if we directly drop all null value without other treatments, we would lose at most 13% of our dataset which is acceptable. The remaining dataset is intact which could be used for building the model.

```
#drop all records with at least 1 na value
df = df.dropna()
```

```
#92.59% remain after drop na
df.info()
print('Percentage of data remain:', round(45221/48840 * 100, 2),'%')
print('Number of records has been dropped out:', 48840 - 45221)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45221 entries, 0 to 48840
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             45221 non-null  int64
 1   Workclass       45221 non-null  object
 2   fnlwgt          45221 non-null  int64
 3   Education       45221 non-null  object
 4   Education_num   45221 non-null  int64
 5   Marital_status  45221 non-null  object
 6   Occupation      45221 non-null  object
 7   Relationship    45221 non-null  object
 8   Race            45221 non-null  object
 9   Sex             45221 non-null  object
 10  Capital_gain    45221 non-null  int64
 11  Capital_loss    45221 non-null  int64
 12  Hours_per_week  45221 non-null  int64
 13  Native_country  45221 non-null  object
 14  Salary          45221 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.5+ MB
Percentage of data remain: 92.59 %
Number of records has been dropped out: 3619
```

As we can see from the screenshot above, after dropped all null records, there are still 45221 records remained which take about 92.59% of the original dataset. Less than 10% of the data had been removed and the remaining data is perfectly intact for making analysis or prediction. Therefore, I choose to directly drop all null values without further treatments.

Part 2: Model Building

To build the KNN model that classify low income from high income citizens, we need to firstly label our observations as either high or low income. There is a column in the dataset called 'Salary' which indicates whether the observation has yearly salary less or equal to $50,000 or greater than $50,000. Based on this column, I created a new column called High/Low as the response variable to categorize observations with salary less or equal to $50,000 as ***Low*** and salary greater than $50,000 as ***High***. Since the KNN model could be able to take numeric variable as the input, we have to convert all categorical variables into numeric variable using ***get_dummies()*** function. After converting categorical variables into numeric variables, we used all attributes as input x for the model to predict y (***High/Low***) the output. Before train the model with data, we need to split the dataset into training set and testing set by 75 and 25 percent. As the data is all prepared, we could start train our model by using the ***KNeighborsClassifier()*** function from sklearn package. Firstly, we need to arbitrarily select k value which is the number of nearest data points used to classify the target point. Then we have to choose the distance metric that used to calculate the distance between each point. Here we choose Euclidean method to calculate the distance and train the model. The k value equals to 40 means we take most common type from the 40 nearest point to classify the target point. After the model had been trained, we test the classification accuracy. We can see that approximately 84.34% of the test set data could be accurately classified by this KNN model.

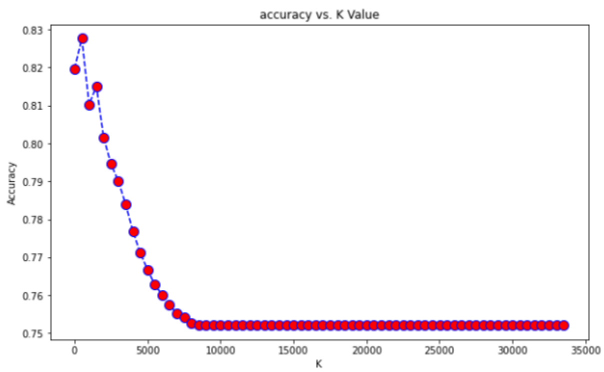
```
k=40
knn = KNeighborsClassifier(n_neighbors=k,metric='euclidean')
knn.fit(X_train,y_train)
```

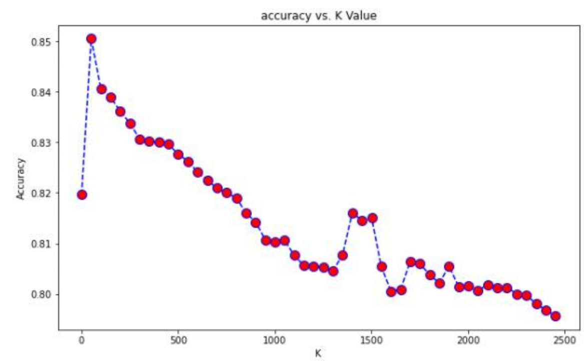
```
#predict type of Iris based on all attributes in the dataset
y_pred = knn.predict(X_test)
```

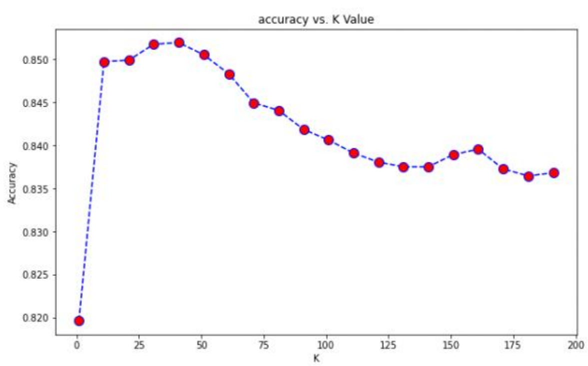
```
print(accuracy_score(y_test,y_pred))
```

```
0.843445957898461
```

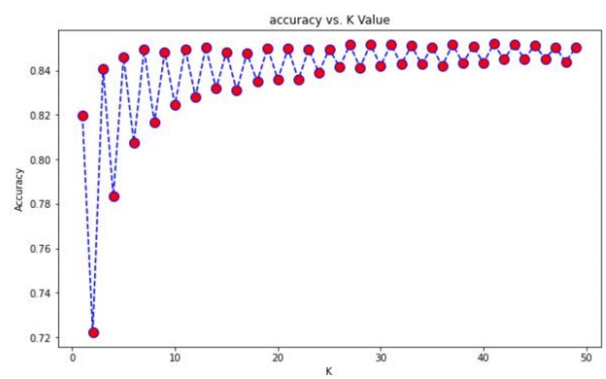
To find the optimal k value that generate the highest accuracy score, I graphed the k-value with the accuracy to visualize the trend. At the beginning, as the k value increases, the accuracy score increases as well. Until the accuracy score reaches to the maximum, the increasing of k-value would accompany with a decreasing of accuracy and stabilized between 0.75 to 0.76.








```
Maximum accuracy: 0.851937024588714 at K = 40
```



As I continue to shrink the range of k value from 0 to 2500, 0 to 200, and 0 to 50. Then I found the k value equals to 40 has the maximum accuracy which is around 0.8519 or 85.19%.

In conclusion, through incorporating all attributes from the dataset to construct the KNN model, and choosing the optimal k value by shrinking the range of k value, we finally made the KNN model with the highest accuracy score of 0.851937024588714 at K = 40, which means that this KNN model could accurately classify around 85% of testing set data (U.S. citizen) according to their attributes provided in the dataset into either high income group or low income group.

Conclusion

For this week's assignment, we firstly clean the dataset and then constructed KNN model to classify observations into high- and low-income group. However , since I incorporated with categorical attributes to predict the high- and low-income group, the dimensionality becomes very large which makes the calculation of distance become each data point become slower and the whole algorithm become very slow when I try to find the optimal K value. One of the possible ways for improvement is to use PCA (principle component analysis) to reduce dimensionality.

Reference:

How to find the optimal value of K in KNN? - Towards Data Science. (2022, April 18). Medium. Retrieved September 29, 2022, from https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb

Adamczyk, J. (2022, March 30). *Make kNN 300 times faster than Scikit-learn’s in 20 lines!* Medium. Retrieved September 29, 2022, from https://towardsdatascience.com/make-knn-300-times-faster-than-scikit-learns-in-20-lines-5e29d74e76bb