Toronto, Canada

**A Report on**

**Module 1 Project**

# Predictive Analytics (ALY 6020)

Guided by:

Prof. Dr. Harpreet Sharma

Submitted By:

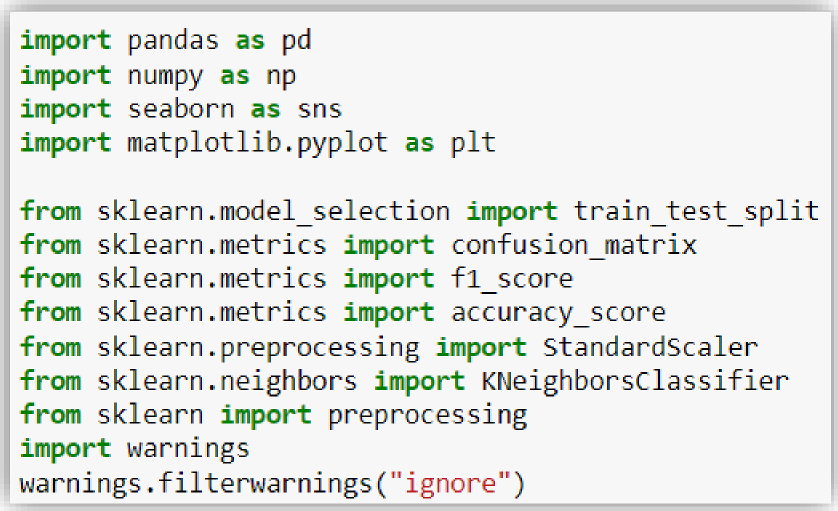
| Name of Student | NUID | Date of submission |
|---|---|---|
| Parth Shah | 002956963 | 20th January, 2023 |

# Appendix

## Introduction

- The US Census Bureau, a branch of the US Department of Commerce, conducts the US Census every ten years.
- Every 10 years, there is a census that is used to gather information about the population and housing in the US.
- The borders of legislative and administrative districts are redrawn using this data, which is also used to redistribute the seats in the US House of Representatives.
- Businesses, academics, and governments all utilize the census data to inform their decisions.
- The information gathered contains details on many different factors, such as age, sex, race, relationship status, education, employment, and income.

## Data Analysis

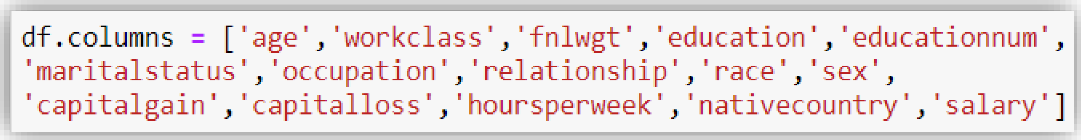


```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")
```

**Figure 1: Importing Libraries**

This code imports several libraries that are commonly used in data science and machine learning projects.

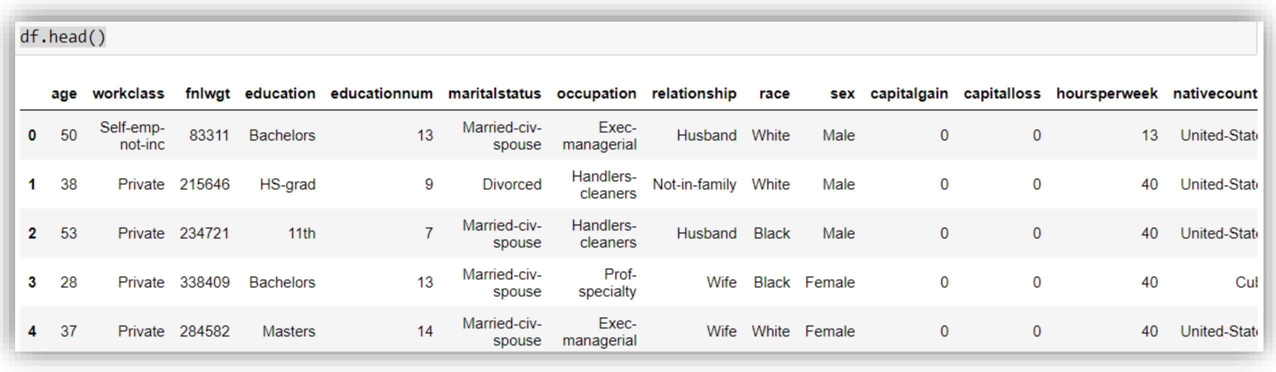
- pandas is used for data manipulation and analysis.
- numpy is used for numerical computations and array operations.
- seaborn and matplotlib.pyplot are used for data visualization.
- sklearn.model_selection is used for splitting the data into training and testing sets.
- sklearn.metrics is used for evaluating the performance of the model, such as calculating the confusion matrix, f1 score, and accuracy score.
- sklearn.preprocessing is used for standardizing the data and normalizing the features.
- sklearn.neighbors is used for implementing the k-nearest neighbors algorithm.
- Warnings are used to filter out warning message that may be generated by the code.
- It is also importing StandardScaler, KNeighborsClassifier, preprocessing from sklearn library.



```
df.columns = ['age','workclass','fnlwgt','education','educationnum',
'maritalstatus','occupation','relationship','race','sex',
'capitalgain','capitalloss','hoursperweek','nativecountry','salary']
```

**Figure 2: Adding name to columns.**

- The columns of a DataFrame called "df" are being renamed by this code.
- The columns' names are being changed to those listed in the list: "age," "workclass," "fnlwgt," "education," "educationnum," "maritalstatus," "occupation," "relationship," "race," "sex," "capitalgain," "capitalloss," "hours per week," "native country," and "salary."
- This is done to ensure that the names of the columns in the DataFrame are clear and relevant.
- It will be simpler to do data analysis and visualization on the data as a result.

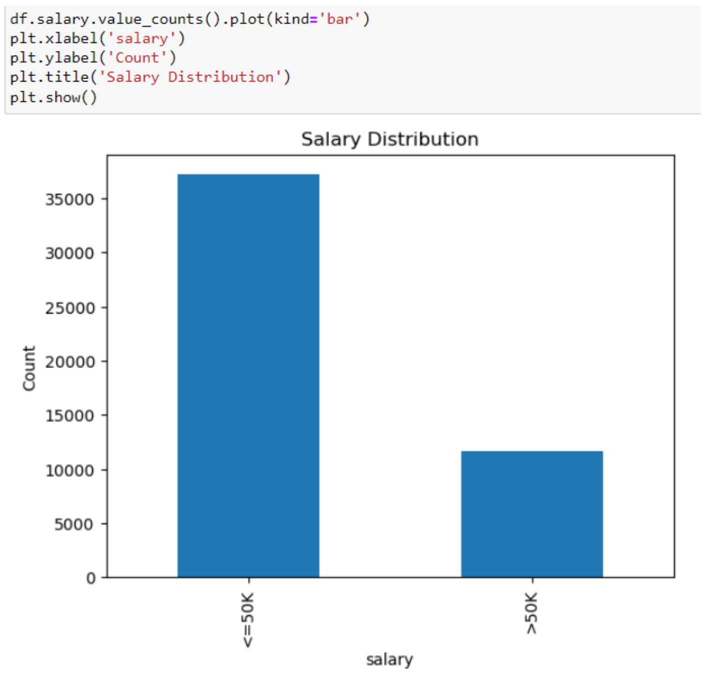


```
df.head()
```

| | age | workclass | fnlwgt | education | educationnum | maritalstatus | occupation | relationship | race | sex | capitalgain | capitalloss | hoursperweek | nativecount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | 13 | United-Stat |
| 1 | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | 40 | United-Stat |
| 2 | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black | Male | 0 | 0 | 40 | United-Stat |
| 3 | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black | Female | 0 | 0 | 40 | Cul |
| 4 | 37 | Private | 284582 | Masters | 14 | Married-civ-spouse | Exec-managerial | Wife | White | Female | 0 | 0 | 40 | United-Stat |

**Figure 3: Top five rows of dataset**

- To display the top few rows of a DataFrame in pandas, use the df.head() function. It is often used to rapidly examine a DataFrame's structure and contents.
- By default, the head () method provides the top 5 rows of the DataFrame, but you may select a different number of rows by giving an integer as a parameter.

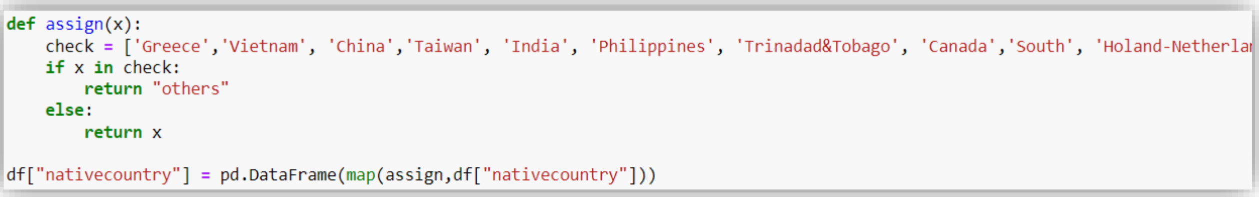


```
df.salary.value_counts().plot(kind='bar')
plt.xlabel('salary')
plt.ylabel('Count')
plt.title('Salary Distribution')
plt.show()
```

**Figure 4: Salary distribution**

- This code is creating a bar plot of the value counts of the "salary" column in a DataFrame called "df". The x-axis is labeled as "salary" and the y-axis is labeled as "Count". The title of the plot is "Salary Distribution". The "plt.show()" function is used to display the plot.

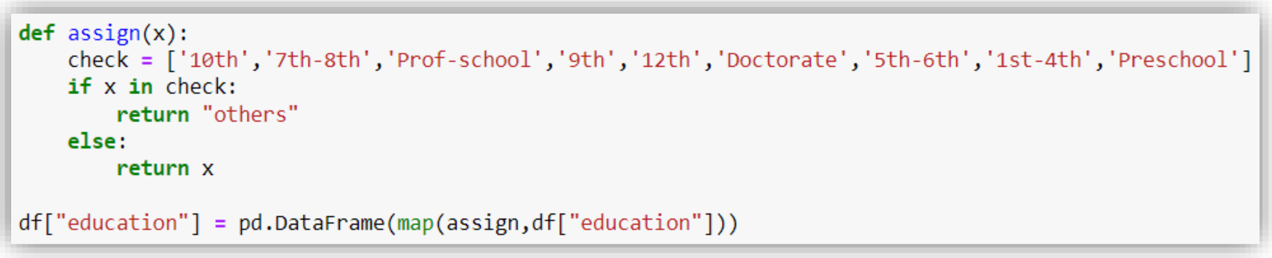
- 10000 persons earn less than $50,000 annually, compared to almost 35000 who earn more than $50,000.



```
def assign(x):
    check = ['Greece','Vietnam', 'China','Taiwan', 'India', 'Philippines', 'Trinadad&Tobago', 'Canada','South', 'Holand-Netherla
    if x in check:
        return "others"
    else:
        return x

df["nativecountry"] = pd.DataFrame(map(assign,df["nativecountry"]))
```

**Figure 5: Assigning value for Native Country**

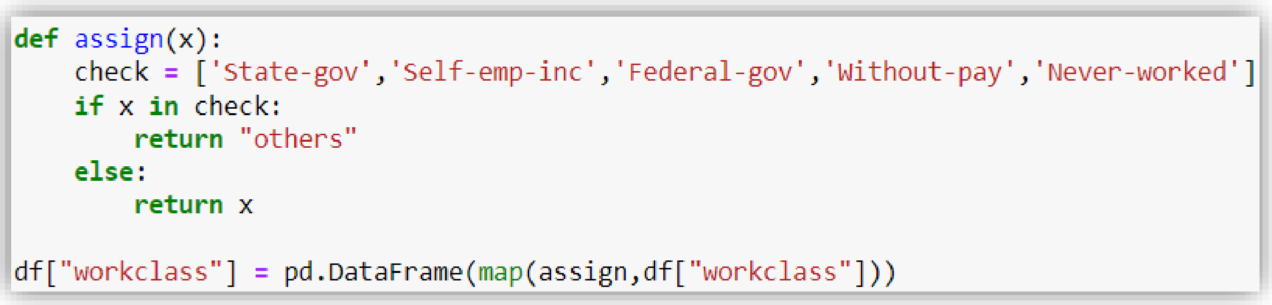
- The "nativecountry" column's value is represented as a string in the dataframe df by the function assign(x), which is defined in this code.
- The function determines whether the value of x is present in the predefined list of nations, which includes Greece, Vietnam, China, Taiwan, India, the Philippines, Trinidad and Tobago, Canada, the South, the Netherlands, Puerto Rico, Poland, Iran, England, Germany, Italy, Japan, Hong, Honduras, Cuba, Ireland, Cambodia, Peru, Nicaragua, El Salvador, Hungary, Columbia, Guatemala, Jamaica, Ecuador, France, and Yugoslavia.
- It returns "others" if the value of x is in the list; else, it returns the value of x.
- The assign() function is then applied to each element in the "nativecountry" column of the DataFrame df using the map() function. The resultant numbers are then placed in a new dataframe, which is subsequently assigned to the dataframe df's "nativecountry" column. This is used to collect all the nations that are not on the function's list into a single category called "others," which is helpful for data analysis and display.



```python
def assign(x):
    check = ['10th','7th-8th','Prof-school','9th','12th','Doctorate','5th-6th','1st-4th','Preschool']
    if x in check:
        return "others"
    else:
        return x

df["education"] = pd.DataFrame(map(assign,df["education"]))
```

**Figure 6: Assigning value for Education.**

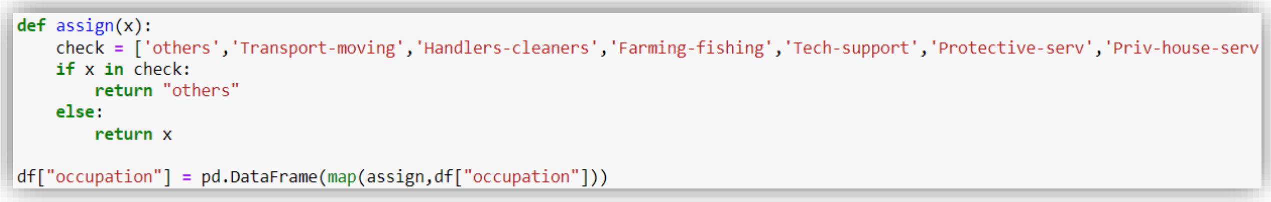
- The "education" column's value is represented as a string in the code that specifies the method assign(x), which only accepts one input, x.
- The function determines if the value of x is included in the preset list of educational levels ["10th," "7th," "Prof-school," "9th," "12th," "Doctorate," "5th," "1st," and "Preschool."
- It returns "others" if the value of x is in the list; else, it returns the value of x.
- The assign() function is then applied to each element in the "education" column of the DataFrame df using the map() function. The resultant numbers are then placed in a new dataframe, which is subsequently allocated to the dataframe df's "education" column.
- This is used to collect any educational levels that are not already included in the function's list into a single category called "others," which is helpful for data analysis and display.



```python
def assign(x):
    check = ['State-gov','Self-emp-inc','Federal-gov','Without-pay','Never-worked']
    if x in check:
        return "others"
    else:
        return x

df["workclass"] = pd.DataFrame(map(assign,df["workclass"]))
```

**Figure 7: Assigning value for Workclass.**

- The "workclass" column's value is represented as a string in the code that specifies the method assign(x), which only accepts one input, x.
- The function determines if the value of x is included in the list of preset workclass levels ["State-gov", "Self-emp-inc", "Federal gov", "Without-pay", and "Never-worked"].
- It returns "others" if the value of x is in the list; else, it returns the value of x.
- The assign() function is then applied to each element in the "workclass" column of the DataFrame df using the map() function. The resultant values are then placed in a new dataframe, which is subsequently assigned to the dataframe df's column "workclass."
- The "others" category is used to aggregate all workclass levels that are not on the list supplied by the function. This is important for data analysis and visualization.



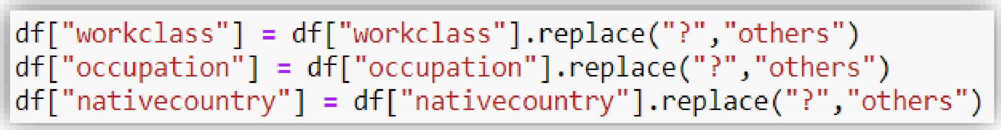
```
def assign(x):
    check = ['others','Transport-moving','Handlers-cleaners','Farming-fishing','Tech-support','Protective-serv','Priv-house-serv
    if x in check:
        return "others"
    else:
        return x

df["occupation"] = pd.DataFrame(map(assign,df["occupation"]))
```

**Figure 8: Assigning value for Occupation.**

- The "occupation" column's value is represented as a string in the code that specifies the method assign(x), which only accepts one input, x.
- The function determines if the value of x is included in the established list of occupation levels ["others," "Transport-moving," "Handlers-cleaners," "Farming," "Tech," "Protective," "Priv," and "Armed Forces."
- It returns "others" if the value of x is in the list; else, it returns the value of x.
- The map() function is then used to apply the assign() function to each element in the "occupation" column of the DataFrame df. The resulting values are then added to a new

dataframe and assigned to the column labelled "occupation" in the dataframe df.

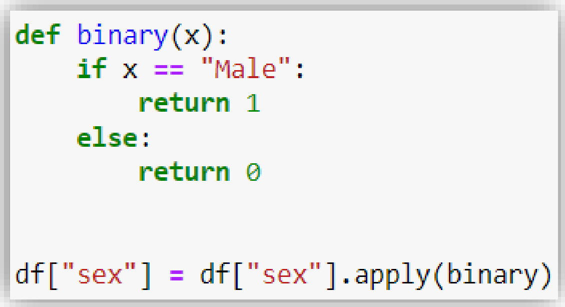
- All profession levels that aren't in the list the function returned are combined under the "others" category. For data processing and visualization, this is crucial.



```
df["workclass"] = df["workclass"].replace("?","others")
df["occupation"] = df["occupation"].replace("?","others")
df["nativecountry"] = df["nativecountry"].replace("?","others")
```

**Figure 9: Replacing Values**

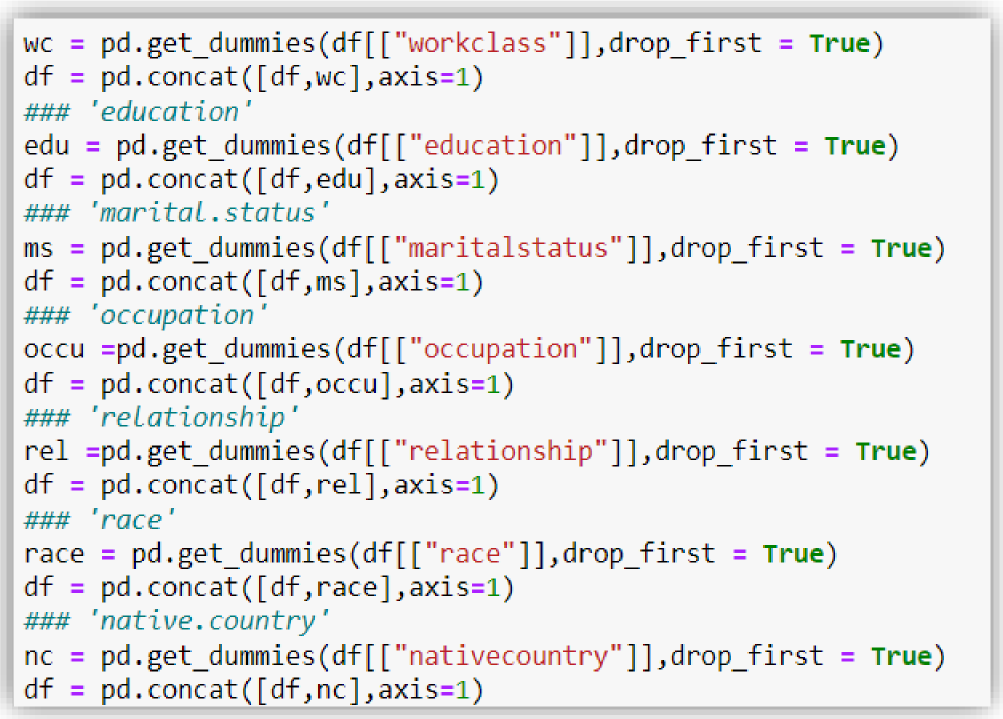
- This code changes all "?" values in the columns for "workclass," "occupation," and "nativecountry" to "others."
- Values in a DataFrame can be changed using the replace() function in pandas. The method accepts two arguments: the value you wish to change and the value with which you wish to replace it.
- The "workclass," "occupation," and "nativecountry" columns of the DataFrame df are called upon in this code to invoke the procedure. It handles missing values in the data by replacing all instances of "?" with "others." When working with real-world datasets, which frequently have missing or damaged data, this is helpful. In data analysis and modelling, it helps to avoid mistakes or unexpected outcomes by substituting missing variables with a certain value.



```
def binary(x):
    if x == "Male":
        return 1
    else:
        return 0


df["sex"] = df["sex"].apply(binary)
```

**Figure 10: Converting Sex to 1 and 0**

- This code defines a function binary(x) that takes in one parameter x, which is a string representing the value of the "sex" column in the dataframe df.
- The function checks if the value of x is "Male"
- If the value of x is "Male" it returns 1, otherwise it returns 0.
- Then, the code uses the apply() function to apply the binary() function to each element in the "sex" column of the DataFrame df. The resulting values are stored in the same column and it replaces the original values.
- This is useful for converting categorical variable into numerical variable. This will be useful in the process of modelling and prediction.
- This function is converting Male to 1 and Female to 0, this will be useful in the modelling process as the model will be able to understand the numerical value instead of categorical value.



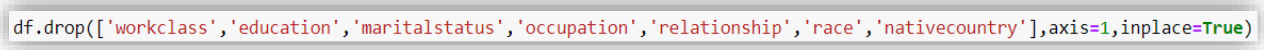
```
wc = pd.get_dummies(df[["workclass"]],drop_first = True)
df = pd.concat([df,wc],axis=1)
### 'education'
edu = pd.get_dummies(df[["education"]],drop_first = True)
df = pd.concat([df,edu],axis=1)
### 'marital.status'
ms = pd.get_dummies(df[["maritalstatus"]],drop_first = True)
df = pd.concat([df,ms],axis=1)
### 'occupation'
occu =pd.get_dummies(df[["occupation"]],drop_first = True)
df = pd.concat([df,occu],axis=1)
### 'relationship'
rel =pd.get_dummies(df[["relationship"]],drop_first = True)
df = pd.concat([df,rel],axis=1)
### 'race'
race = pd.get_dummies(df[["race"]],drop_first = True)
df = pd.concat([df,race],axis=1)
### 'native.country'
nc = pd.get_dummies(df[["nativecountry"]],drop_first = True)
df = pd.concat([df,nc],axis=1)
```

**Figure 11: Creating Dummy Variable**

- This code is using the get_dummies() function in pandas to convert categorical variables into numerical variables.
- For each categorical variable in the dataset, the function creates a new DataFrame with one column for each category and assigns a value of 1 or 0 to indicate whether the original value belonged to that category.
- For each categorical variable "workclass", "education", "maritalstatus", "occupation", "relationship", "race" and "nativecountry", it is creating a new DataFrame using get_dummies and then concatenating the original dataframe with the newly created dataframe using pd.concat() function with axis = 1, which concatenates dataframe along the columns.
- The drop_first parameter is set to True so that the first column for each categorical variable

is dropped, this is to avoid multicollinearity in the data.

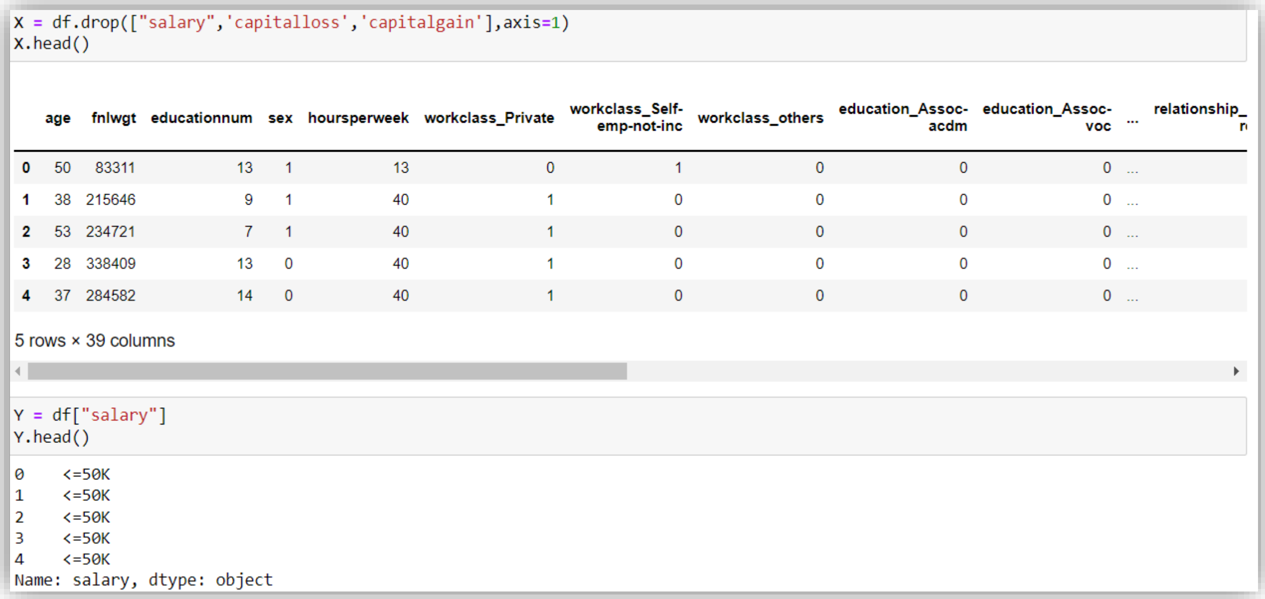
- This is useful for creating numerical variables from categorical variables, and it is useful in the process of modeling and prediction.



```
df.drop(['workclass','education','maritalstatus','occupation','relationship','race','nativecountry'],axis=1,inplace=True)
```

**Figure 12: Dropping values.**

- The columns "workclass," "education," "marital status," "occupation," "relationship," "race," and "native country" are being removed from the DataFrame df by this code.
- A DataFrame's rows or columns can have specific labels removed using the pandas method drop().



```
X = df.drop(["salary",'capitalloss','capitalgain'],axis=1)
X.head()
```

| | age | fnlwgt | educationnum | sex | hoursperweek | workclass_Private | workclass_Self-emp-not-inc | workclass_others | education_Assoc-acdm | education_Assoc-voc | ... | relationship_ r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 50 | 83311 | 13 | 1 | 13 | 0 | 1 | 0 | 0 | 0 | ... | |
| **1** | 38 | 215646 | 9 | 1 | 40 | 1 | 0 | 0 | 0 | 0 | ... | |
| **2** | 53 | 234721 | 7 | 1 | 40 | 1 | 0 | 0 | 0 | 0 | ... | |
| **3** | 28 | 338409 | 13 | 0 | 40 | 1 | 0 | 0 | 0 | 0 | ... | |
| **4** | 37 | 284582 | 14 | 0 | 40 | 1 | 0 | 0 | 0 | 0 | ... | |

5 rows × 39 columns

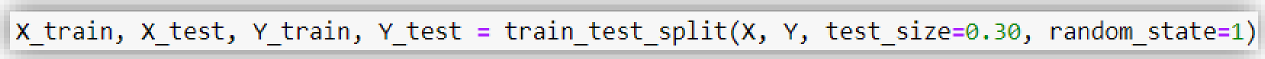
```
Y = df["salary"]
Y.head()
```

```
0    <=50K
1    <=50K
2    <=50K
3    <=50K
4    <=50K
Name: salary, dtype: object
```

**Figure 13: Creating Variables X and Y**

- Two new variables, X and Y, are being created by this code from the DataFrame df.
- The columns "salary," "capitalloss," and "capitalgain" are removed from the DataFrame using the drop() method, and the remaining DataFrame is assigned to the variable X.

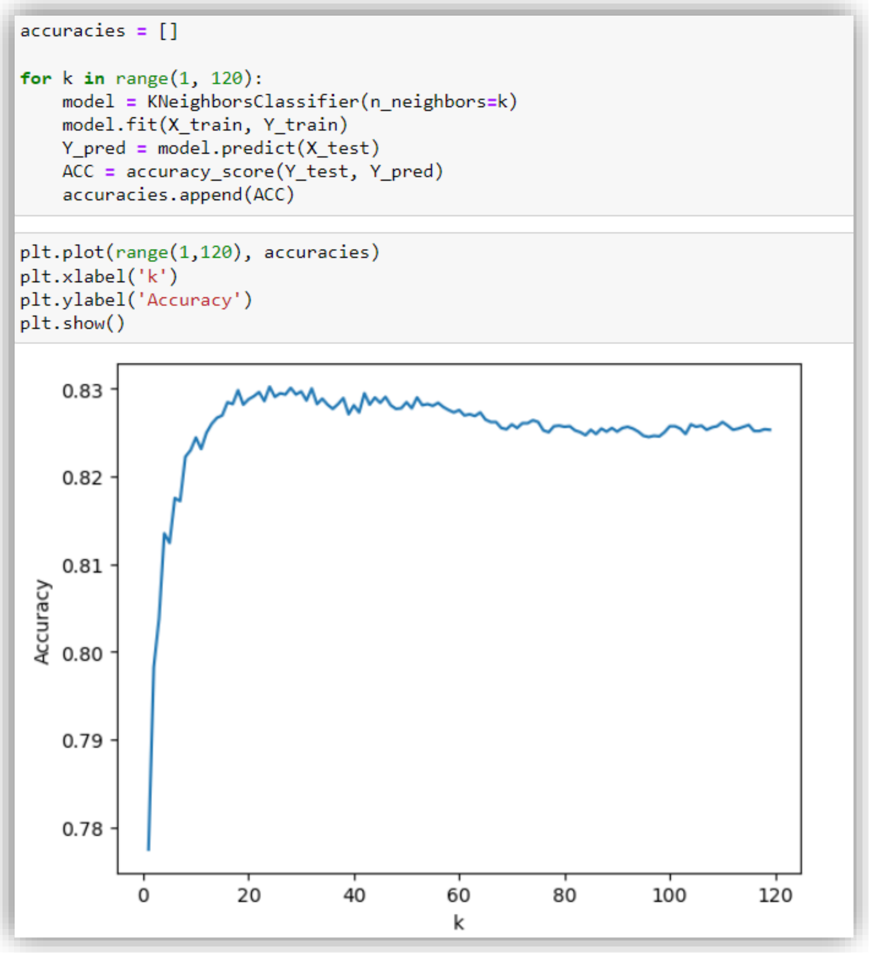
- Additionally, it is generating a new variable Y, which represents the original DataFrame's "salary" column.
- Now that X and Y are distinct, where X stands for the feature variables and Y for the target variables.
- This distinguishes the input variables (X) from the output variable, which is helpful for training and evaluating machine learning models (Y).
- As the machine learning models are trained on the input variables before being used to predict the target variable, this separation is required.



```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=1)
```

**Figure 14: Splitting dataset.**

- The data is divided into training and testing sets using the train test split() method from the sklearn.model selection package.
- The X train, X test, Y train, and Y test objects are the four that the function returns. The feature variables and target variables for the training set are X train and Y train, respectively. The feature variables and target variables for the test set are X test and Y test, respectively.



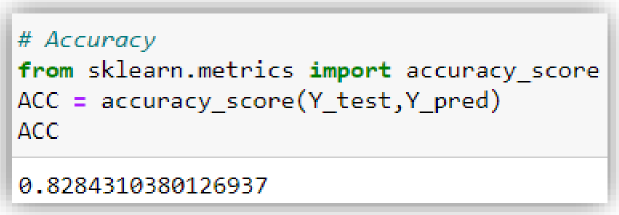
```
accuracies = []

for k in range(1, 120):
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    ACC = accuracy_score(Y_test, Y_pred)
    accuracies.append(ACC)
```

```
plt.plot(range(1,120), accuracies)
plt.xlabel('k')
plt.ylabel('Accuracy')
plt.show()
```

**Figure 15: Elbow Method and plot**

- It begins by initializing a list named accuracies, which is empty.
- It then begins a for loop that loops through the numbers between 1 and 120.
- The number of nearest neighbours to take into account is determined by the current value of the loop variable k, and a new instance of the KNeighborsClassifier class is produced for each iteration of the loop.
- The accuracy score() function from the sklearn.metrics package is then used to calculate the model's accuracy on the test set. This method accepts the real test labels (Y test) and the predicted labels (Y pred) as parameters.

- After then, the accuracy is added to the list of accuracy.

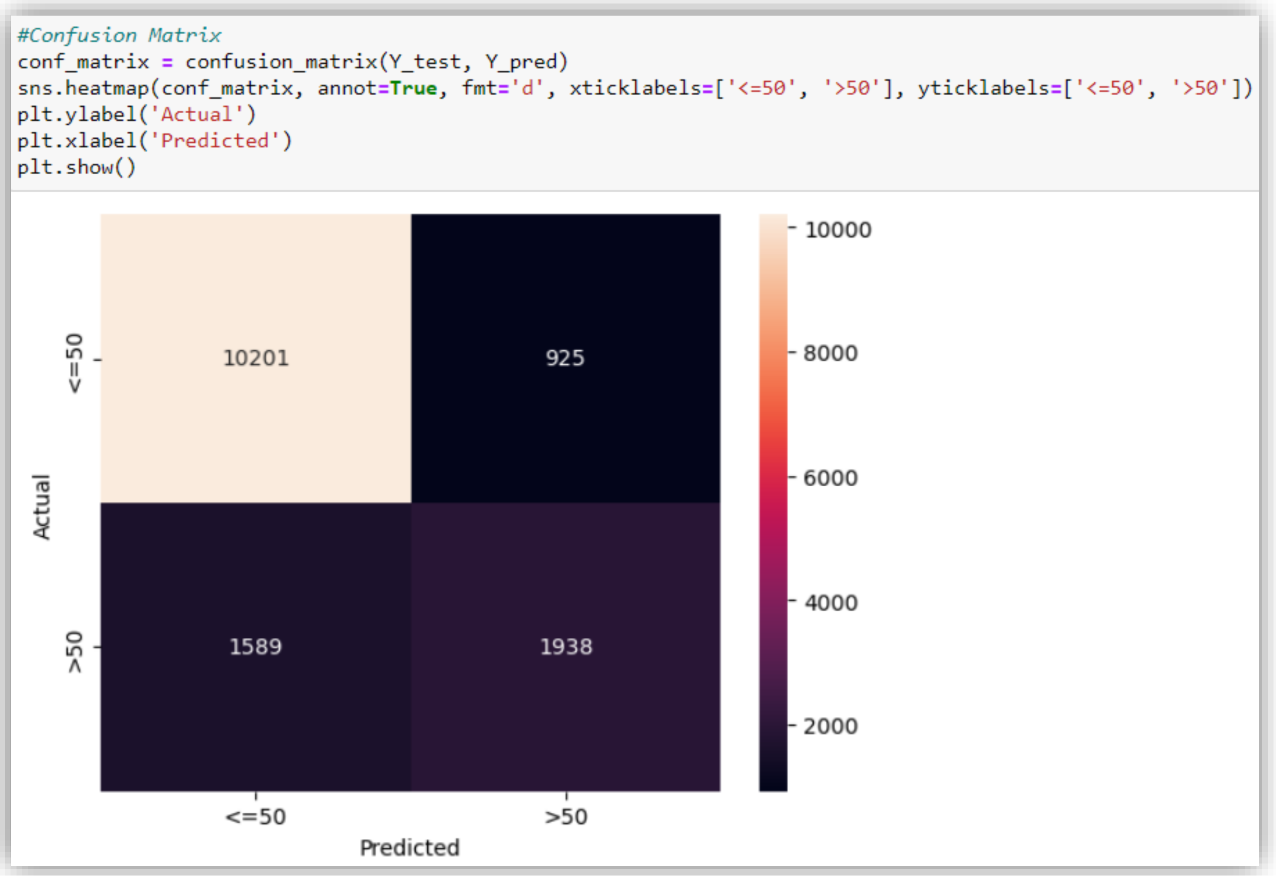


```
# Accuracy
from sklearn.metrics import accuracy_score
ACC = accuracy_score(Y_test,Y_pred)
ACC
```

```
0.8284310380126937
```

**Figure 16: Accuracy of the model**

- The accuracy score() function from the sklearn.metrics package is used in this code to calculate the model's accuracy on the test set.
- The real test labels (Y test) and the anticipated labels (Y pred) are the two inputs that the function requires.
- The accuracy score is then calculated based on the percentage of cases that were properly categorized.
- Accuracy of the model is 83 percent.



```
#Confusion Matrix
conf_matrix = confusion_matrix(Y_test, Y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', xticklabels=['<=50', '>50'], yticklabels=['<=50', '>50'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

**Figure 17: Confusion Matrix**

- The confusion matrix for the model's predictions on the test set is computed by this code using the confusion matrix() function from the sklearn.metrics package.
- The real test labels (Y test) and the anticipated labels (Y pred) are the two inputs that the function requires.
- A table called the confusion matrix is used to describe how well a classification model performs. The dataset has n classes, hence the matrix is a nxn matrix. The matrix provides information on the number of accurate forecasts, false positives, false negatives, true positives, and wrong predictions.

## Reference

- Python Machine Learning - Confusion Matrix. (n.d.). Python Machine Learning - Confusion Matrix. Retrieved January 20, 2023, from https://www.w3schools.com/python/python_ml_confusion_matrix.asp#:~:text=What%20is%20a%20confusion%20matrix,the%20predictions%20we%20have%20made
- Elbow Method for optimal value of k in KMeans - GeeksforGeeks. (2019, June 6). GeeksforGeeks. Retrieved January 20, 2023, from https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/