Lab 5 Seminar Activity 1

Outlier detection

Outlier detection is the process of identifying data points that are significantly different from the rest of the data in a dataset. Outliers are observations that deviate from the majority of the data, either in terms of their values or their distribution.

Outliers can occur for a variety of reasons, such as measurement errors, data entry errors, or genuine deviations from the norm. Outlier detection is important in many fields, including finance, healthcare, and engineering, where outliers can have a significant impact on the accuracy and reliability of statistical analyses and machine learning models.

Please import all the libraries that we have used so far

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Please load the adult.csv dataset

```
In [2]: Adult_data = pd.read_csv('adult.csv')
```

```
In [ ]:
```

Please use all the functions like .head(), .tail() and so on which you have done so far in previous seminar lab activities

```
In [ ]:
```

As you can see there are '?' in three columns Occupation, workclass and native_country, replace those '?' in the dataframe with nan and use the .isnull().sum() function to count the numver of missing values.

```
In [ ]:
```

```
In [ ]:
```

Fill the values in a numerical column with mean value

```
In [ ]:
```

```
In [ ]:
```

Replace the missing value in a categorical column with most frequent values, use the mode()

```
In [ ]:
```

```
In [ ]:
```

Start working with Outlier detection

Calculate the Z score using Python to find this outlier.

In [12]:
```
#calculate mean, standard deviation

data = [1, 2, 2, 2, 3, 17, 1, 1, 4, 5, 4, 3, 2, 2, 2, 3, 1, 1, 2]
mean = np.mean(data)
std = np.std(data)
print('mean of the dataset is', mean)
print('std. deviation is', std)
```

```
mean of the dataset is 3.0526315789473686
std. deviation is 3.4712910416507685
```

In [13]:
```
#calculate z score. If z score > 3, print it as an outlier

threshold = 3
outlier = []
for i in data:
    z = (i-mean)/std
    if z > threshold:
        outlier.append(i)
print('outlier in dataset is', outlier)
```

```
outlier in dataset is [17]
```

From the above lines of code, we can see that Z score helps us identify outliers in the data.

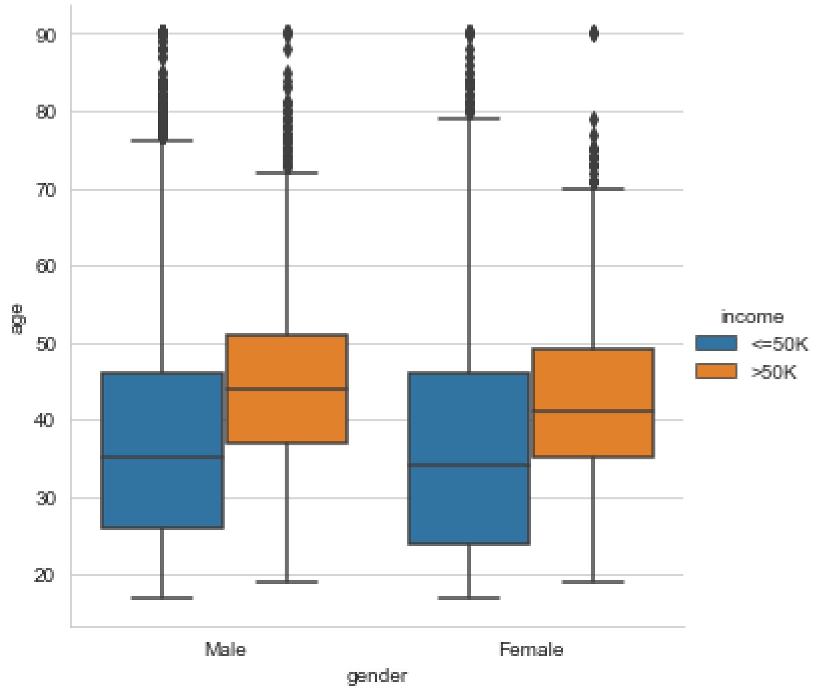
Let's take an example of boxplot to detect the outliers

We start now looking at outliers. For the purpose of looking at outliers, let us consider the continous columns we have already defined only so X can be equal to the CONTINUOUS_COLUMNS of the data frame. We can create a new train_X wich we can call train_OL_X with the CONTINUOUS_COLUMNS only.

In [5]:
```
#box plot to see the outliers

sns.set_style("whitegrid")
sns.catplot(x='gender', y='age',hue="income",  kind="box", data=Adult_data)
```

Out[5]:
```
<seaborn.axisgrid.FacetGrid at 0x1d68fdee940>
```



In [6]:
```
#Designate the continuous input features as train_OL_X
CONTINUOUS_COLUMNS = ["age", "educational-num", "capital-gain", "capital-loss",
                      "hours-per-week","income"]

X= Adult_data[CONTINUOUS_COLUMNS]

train_X=X.dropna()#Missing values needs to be hndled before outlier detection can be performed
train_y=train_X['income']
train_OL_X=train_X.drop('income',axis=1)
X.shape
```

Out[6]:
```
(48842, 6)
```

Outlier detection: DBSCAN

We now try to detect outliers, first with the DBSCAN algorithm. Since this file is rather large we do not print the objects with their allocation (outliers designated as -1, or not outliers) but we can print the total number of outliers found. We can also alter the parameters `min_samples` and `eps` to see the effect on the outliers detected. Once you have the code working, experiment with the algorithm parameters to get a not too large number of outliers. We can apply this on the train data only, but to the one with continous columns, i.e. train_OL_X.

In [7]:
```
#import the implementation of this algorihm from sklearn
from sklearn.cluster import DBSCAN

#Use the algorithm for outlier detection, the retun in clusters will show the membership of each point
#Any point labelled as -1 is an outlier

outlier_detection = DBSCAN(min_samples = 3, eps = 10)
clusters = outlier_detection.fit_predict(train_OL_X)

#Count total number of outliers as count of those labelled as -1
TotalOutliers=list(clusters).count(-1)
#print (clusters)
print("Total number of outliers identified is: ",TotalOutliers)
```

```
Total number of outliers identified is:  653
```

We can now create a mask or filter to ensure only those rows that are not outliers are retained in a new data frame that we can later use for classification. Let us create a new output variable *y1* and input set of variables *X1* which contain a filtered version of the original data frame. For this, we can create a mask which takes the value of `clusters!= -1` . This will be a boolean array which we can then use to filter *y* into a new version *y1*, and similarly *X* into a new version *X1*. Check the shape of the new X and y with `.shape` to see the size of each. The amount of rows should be equal to the rows in the original data frame minus the rows that were designated as outliers. Note that we need to filter the data frame that contains all the columns (*train_X, train_y*), and not just the numeric ones, as all columns will be needed for the classification algorithms.

In [8]:
```
# select all rows that are not outliers and create a boolean mask
mask = clusters != -1
# Apply mask to y and check shape
y1= train_y[mask]
print (y1.shape)

#Apply mask to X and check shape
X1=train_X[mask]
print(X1.shape)
```

```
(48189,)
(48189, 6)
```

If you wish to save any of the dataframes you have created to load them elsewhere you can do that with the `.to_csv()` method. You can pass inside as parameters the path and file name and `index = False` if you don't wish to save the index. Alternatively, you can repeat the code above to get the data frame in a later lab.

In [9]:
```
#Save X1 to a file for later use if necessary
X1.to_csv("X1.csv", index = False)
```

Let us now do similarly but using the `IsolationForest` algorithm. Again, investigate the parameters to understand how many outliers are found as we change those paramaters.

In [10]:
```
#import the implementation of this algorithm from sklearn
from sklearn.ensemble import IsolationForest

#Use the algorithm for outlier detection, then use it to predict each point
#Any point labelled as -1 is an outlier
clf = IsolationForest(max_samples=48149, random_state = 1, contamination= 0.01)
preds = clf.fit_predict(train_OL_X)
#print(preds)
totalOutliers=0
for pred in preds:
    if pred == -1:
        totalOutliers=totalOutliers+1
print("Total number of outliers identified is: ",totalOutliers)
```

```
C:\Users\44758\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, bu
t IsolationForest was fitted with feature names
  warnings.warn(

Total number of outliers identified is:  489
```

Again, we can create a mask or filter to ensure only those rows that are not outliers are retained in a new data frame that we can later use for classification. Let us create a new output variable *y2* and input set of variables *X2* which contain a filtered version of the original data frame, this time with the isolation algorithm filter. For this, we can create a mask which takes the value of `preds!= -1` . This will be a boolean array which we can then use to filter *train_y* into a new version *y2*, and similarly *Train_X* into a new version *X2*. Again check the shape of the new X and y with `.shape` to see the size of each.

In [11]:
```
# select all rows that are not outliers and create a boolean mask
mask = preds != -1
# Apply mask to y and check shape
y2= train_y[mask]
print (y2.shape)

#Apply mask to X and check shape
X2=train_X[mask]
print(X2.shape)
```

```
(48353,)
(48353, 6)
```

Save X2 if you want!

Now that is all for this activity!!! We have created a number of data frames we may use in later labs for futher analysis such as classification and clustering so make sure you save your work ready for re-use later.