Experiment No 12

* 1. **Aim/Purpose of the Experiment**

To familiarize the students with model evaluation methods using evaluation matrices

**Learning Outcomes**

Knowledge of the model evaluation methods using different matrices.

* 1. **Prerequisites**

Basic knowledge of programming, python syntax, matplotlib, seaborn, different libraries.

* 1. **Materials/Equipment/Apparatus / Devices/Software required**

Jupyter Notebook.

* 1. **Introduction and Theory**
* **Model evaluation** is an important part of a training process since it ensures that a model has been properly trained on the existing dataset and can be used to make predictions for an unseen dataset.
* There are a number of metrics such as accuracy, precision, recall, f1-score that are commonly used to evaluate a model. They are calculated based on the number of correct and incorrect predictions broken down by each class.

#Importing libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Read the dataset

df=pd.read\_csv('metrics\_dataset.csv')

df.head()

df.info()

df['Donated'].value\_counts()

print ('# duplicates label 0:', len(df[(df['Donated']==0) & (df.duplicated())]))

print('# duplicates label 1:',len(df[(df['Donated']==1) & (df.duplicated())]))

# drop duplicates

df= df.drop\_duplicates()

# slightly imbalanced dataset

print('% of label 1 samples: {:.2%}'.format(len(df[df['Donated']==1])/len(df)))

print('% of label 0 samples: {:.2%}'.format(len(df[df['Donated']==0])/len(df)))

sns.pairplot(df, hue='Donated', diag\_kind='kde')

sns.pairplot(df, hue='Donated', diag\_kind='kde')

# suspicious correlation. Monetary equals blood donated each time (cc) \* Frequency .

(df['Monetary']/df['Frequency']).unique()

# repeated variable must be removed

df=df.drop('Monetary', axis=1)

print('Number of outlier Recency values: ',(df['Recency']>30).sum())

# removes rows with outlier recency data

df= df[df['Recency']<30]

# generates new feature, normalizes frequency by number of months since the first donation

df['Freq\_month']=df['Frequency']/df['Time']

df.head()

fig, axes= plt.subplots(2,2, figsize=(16,5))

axes=axes.flatten()

for i, col in enumerate (df.drop('Donated', axis=1)):

sns.histplot(df.loc[:,col], ax=axes[i], bins=20, binrange=[min(df[col]),max(df[col])], label='Did Not Donated')

sns.histplot(df.loc[df['Donated']==1,col], ax=axes[i], color='red', bins=20, binrange=[min(df[col]),max(df[col])], label='Donated')

axes[i].legend()

plt.tight\_layout()

* Those who recently donated (low Recency), most likely donate again
* Those with larger Recency are less likely to donate (change in physical/health condition, age, or forgetting)
* Among those with higher donation per month (more active donors), the possibility of donation is higher
* Frequency is less than 10 times for most people which means people are not willing to donate more than 10 times

fig, axes= plt.subplots(1,4, figsize=(16,5))

for i, col in enumerate(df.drop('Donated', axis=1)):

sns.boxplot(x='Donated',y=col, data=df, palette="Set2", order=[0, 1], ax=axes[i], width=0.6)

plt.tight\_layout()

* Recency and Freq\_month are two variables with slightly different median and range of values for two classes

# creates predictors and target datasets

X= df.drop('Donated', axis=1)

y=df['Donated']

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

# split the data into training and test sets

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X, y, test\_size=0.25, stratify=y, random\_state=100)

# scale the predictor features based on the training data

scaler=StandardScaler()

scaler.fit(X\_train)

X\_train=scaler.transform(X\_train)

X\_test=scaler.transform(X\_test)

*# fits the model- KNN was used as it is less impacted by imbalnce in the data compared to other classifiers*

*knn=KNeighborsClassifier(n\_neighbors=7)*

*knn.fit(X\_train, y\_train)*

*# confusion matrix*

*from sklearn.metrics import confusion\_matrix*

*confusion\_matrix(y\_test, knn.predict(X\_test))*

***# oversampling the data***

***smt = SMOTETomek(sampling\_strategy='auto', random\_state=105)***

***X\_train\_smt, y\_train\_smt = smt.fit\_resample(X\_train, y\_train)***

***confusion\_matrix(y\_test, svc.predict(X\_test))***

Please do by yourself.

(Sensitivity, Specificity, Precision, Recall, TPR, FPR)

* 1. **Operating Procedure**
* Open Jupyter note book
* Take a new python file
* Type the code
* Run it
* Take inputs from user
* Observe the results
* Verify the results manually
* Store the note book file
  1. **Precautions and/or Troubleshooting**

**Precautions:**

* Save Your Work: Regularly save your Jupyter Notebook to avoid losing your work. You can save your notebook by clicking on the save icon or using the keyboard shortcut Ctrl + S (or Cmd + S on Mac).
* Restart Kernel: If you encounter unexpected behavior or errors, try restarting the kernel. This clears all the variables and imported modules, essentially resetting the notebook's state. You can restart the kernel by going to the "Kernel" menu and selecting "Restart."
* Clear Outputs: To reduce clutter and confusion, consider clearing the outputs of code cells that are no longer relevant. You can do this by selecting "Clear Outputs" from the "Edit" menu.
* Readability: Keep your code and comments clear and well-organized to make it easier to understand and maintain. Use markdown cells for explanations, headings, and documentation.
* Check Dependencies: If you're using external libraries or packages, ensure they are properly installed in your Jupyter environment. You can check the installed packages by running !pip list or !conda list in a code cell.
* Kernel Selection: Make sure you're using the correct kernel for your notebook. The kernel determines the programming language and environment in which your code runs. You can change the kernel by clicking on "Kernel" > "Change kernel" in the menu.
* Resource Usage: Be mindful of the resources your notebook is using, especially if you're working with large datasets or running intensive computations. Check system monitor tools to ensure you're not exhausting memory or CPU resources.

**Troubleshooting:**

* Syntax Errors: Check for syntax errors in your code. Python is sensitive to indentation and syntax, so ensure your code is properly formatted.
* Variable Scope: Be aware of variable scope issues, especially if you're reusing variable names or working with nested functions.
* Library Installation: If you encounter Module Not Found Error or similar errors, ensure that the required libraries are installed in your Jupyter environment. You can install libraries using !pip install <library> or !conda install <library> in a code cell.
* Kernel Crashes: If the kernel crashes frequently, consider reducing the complexity of your code or optimizing resource usage. Large datasets or intensive computations can sometimes overwhelm the kernel.
* Browser Issues: If you experience rendering or responsiveness issues in the notebook interface, try clearing your browser cache or using a different browser.
* Documentation: Consult the official Jupyter documentation and community forums for additional troubleshooting tips and solutions to common problems.
  1. **Observations**

Observe the results obtained in each operation.

* 1. **Calculations & Analysis**

Calculations should be given for each operation.

* 1. **Result & Interpretation**

Result should be printed and pasted in laboratory copy found from Jupyter note book.

* 1. **Follow-up Questions**
  + What is K means Clustering Algorithm?
  + Is Feature Scaling required for the K means Algorithm?
  + Why do you prefer Euclidean distance over Manhattan distance in the K means Algorithm?
  + Which metrics can you use to find the accuracy of the K means Algorithm?
  + What is a centroid point in K means Clustering?
  + Does centroid initialization affect K means Algorithm?
  + What are the advantages and disadvantages of the K means Algorithm?
  1. **Extension and Follow-up Activities (if applicable)**

NA

* 1. **Assessments**
  2. **Suggested reading**

NA