Hugh Hoang 1833106

Dr. Eick

COSC 3337 *"Data Science I"* Fall 2022
Problem Set1

Last Updated: September 27, 8p

Task 2: Creating a Pipeline Encapsulating Data Sampling, Data Splitting, Feature Selection, Feature Creation and Classification Steps.

Task2 Due: Saturday, Oct. 1, 11:59p
Responsible TA: Navid
Total points: 45

For this assignment, you must use dataset "PS1-Task2.xlsx" uploaded in MS Teams under "Datasets and Code" channel. This dataset has been created synthetically by the TA for learning purposes! This dataset has 8400 samples, and each sample has four features (feature1, feature2, feature 3 and feature 4) presented in first four columns. The fifth column of the dataset indicates the label of each sample. There are two classes indicated by label 0 and 1. Write your code in **Python** or other language you prefer to answer the tasks listed below.

Note: *The colors you choose for your plots must be based on your* ***student ID****. As the dataset includes two labels, you will need two colors. Suppose your student ID is "1234567". You will use the six right-most digits to define the first color in hex format. So, the first color is "#234567" in this example. For second color, you must subtract first color number from "FFFFFF". For your convenience, you can use following function code written in Python to generate your colors:*

```
def plot_colors(student_id):
    color1 = "#"+student_id[1:]
    color2 = "#"+str(hex( int("FFFFFF" ,16) - int(student_id[1:],16)))[2:]
    return color1 , color2
```

Usage example:

```
psid = "1234567"
color1,color2 = plot_colors(psid)
```

Learning objectives:

- ✔ Creating a pipeline
- ✔ Sampling technique
- ✔ Splitting data
- ✔ Data visualization
- ✔ Feature selection
- ✔ Feature creation
- ✔ Classification

Tasks:

2.1. Find the proportion of two class samples. R eport number of class-0 and class-1 samples and the ratio $\frac{\#Class\ 0}{\#Class\ 1}$. **2 Points**

Class-0 is 5000

Class-1 is 3400

Proportion is 5000:3400

Ratio is 5000/3400

2.2. **Sampling (regular):** Write a function that randomly samples "q" number of samples from dataset (without replacement) and return the new created dataset. Set $q = 1000$ and report the number of class-0 and class-1 samples and the ratio $\frac{\#Class\ 0}{\#Class\ 1}$ for new created dataset. (Call this dataset dataset2). **5 Points**

Class-0 is 576

Class-1 is 424

Ratio is 576/424

2.3. **Sampling (Stratified):** Write a function that randomly samples "q" number of samples from dataset (without replacement) and preserves proportion of the number of different class samples. This sampling is called stratified sampling. Set $q = 1000$ and report the number of class-0 and class-1 samples and the ratio $\frac{\#Class\ 0}{\#Class\ 1}$ for new created dataset. (Call this dataset dataset3). **5 Points**

Class-0 is 595

Class-1 is 405

Ratio is 595/405

2.4. **Feature Selection:** Compute the covariance matrix for dataset3. Report this covariance matrix. Select two features that you think they may provide better discrimination between two classes. Report selected features (Feature #1, #2, #3 or #4) and explain your reasons. Create a new dataset including only these two features. Call this dataset dataset4. Write a function for this step. **4 Points**

	Feature 1	**Feature 2**	**Feature 3**	**Feature 4**	**Label**
Feature 1	**219.199369**	**-6.418587**	**-19.754699**	**-2186.927521**	**3.118108**
Feature 2	**-6.418587**	**218.116325**	**652.939398**	**63.638255**	**0.201638**

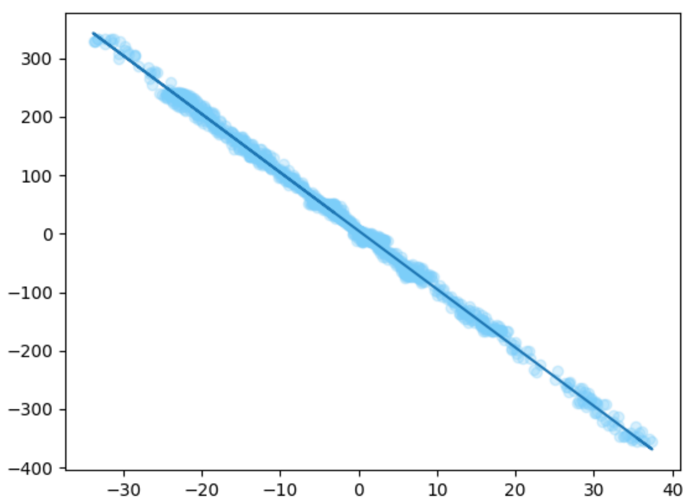
Feature 3   -19.754699 652.939398 1960.982978   195.445080   0.653956

Feature 4 -2186.927521   63.638255   195.445080 21883.777596 -31.146393

Label   3.118108   0.201638   0.653956   -31.146393   0.241216

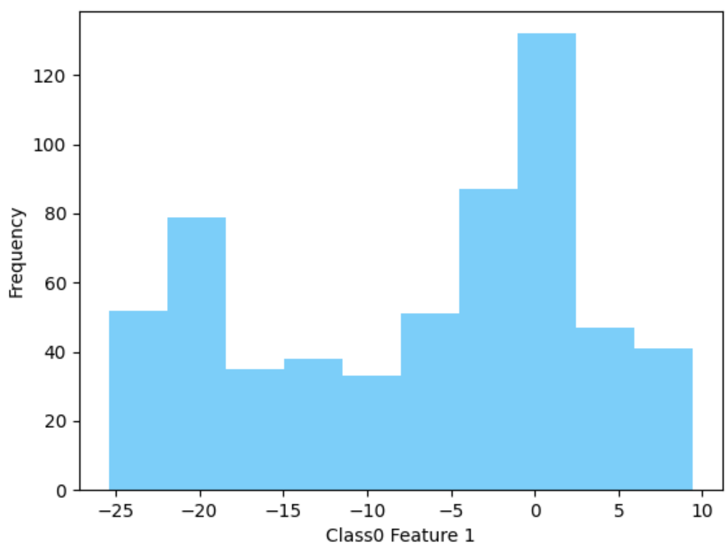
The highest magnitude for a covariance except for the covariance of a variable and itself is -2186 which is feature 1 and 4. Covariance gives you a positive number if the variables are positively related or a negative number if they are negatively related. A high covariance indicates there is a strong relationship between the variables and a low value means there is a weak relationship. Using this since its negative it means they are negatively related and since it has a high covariance there is a strong relationship between the variables, which would make these wo features the best way to discriminate between the two classes.

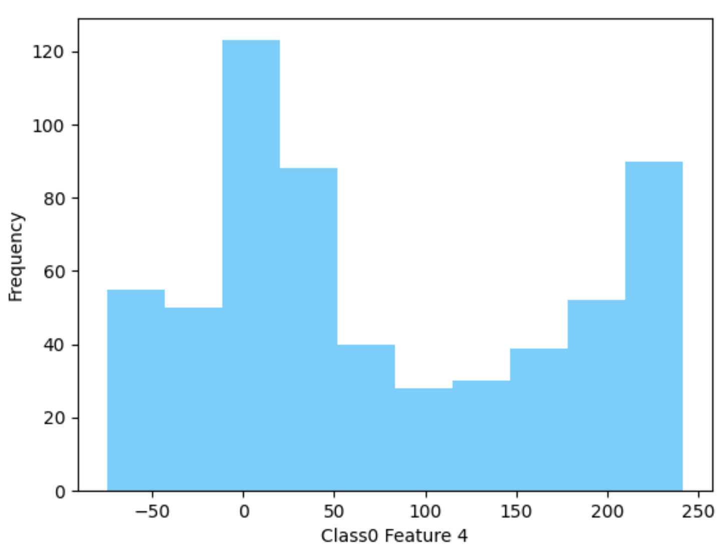
2.5. **Visualization:** Obtain the supervised scatter plot for dataset4. Remember to use your personalized colors for two classes! Do not forget to adjust alpha value (transparency) to see the overlapping areas. Interpret the scatter plot. **2 Points**

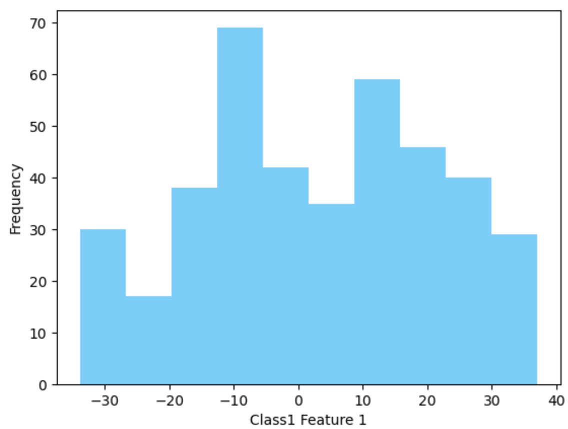


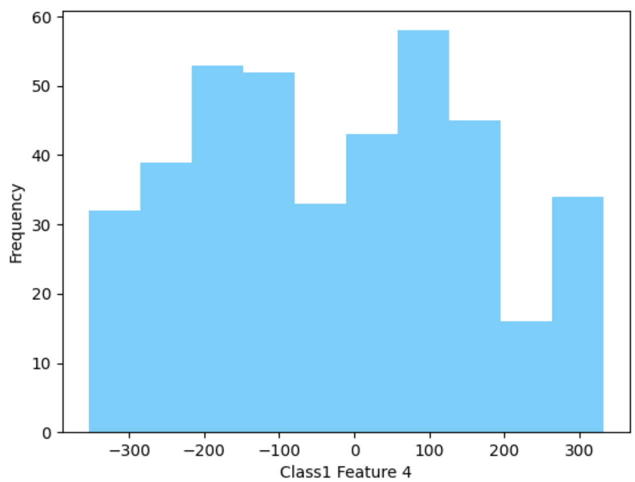
Just like what we predicted for problem 2.5, since the covariance is negative it means the points will be negatively related and since it has a high covariance there is a strong relationship between the variables as we can see with a negative regression line. We can also tell that the correlation for the points must be relatively high as all the points tend to be very close to the line of regression.

2.6. **Visualization:** Obtain four histograms, one for each selected feature in dataset4 and each class instances (first selected feature for class 0 instances, first selected feature for class 1 instances, second selected feature for class 0 instances and second selected feature for class 1 instances). Remember to use your personalized colors for two classes! Discuss the difficulty of separating two classes based on the selected features. **3 Points**





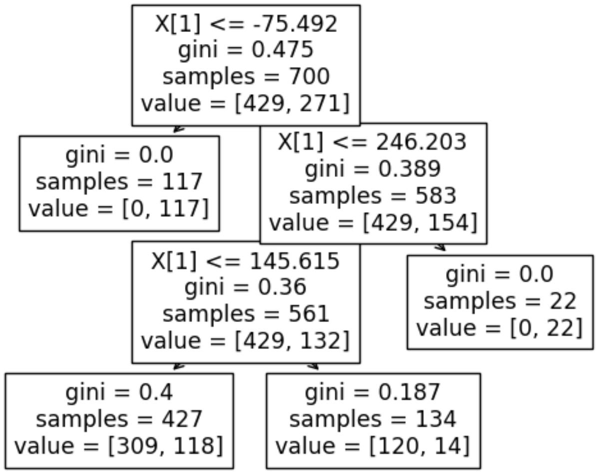




I would say that separating the two classes based on the selected features was not too difficult. It only required me to separate database3 into two groups, each of them by their separate labels and then afterwards sort them based on the two features which in this case is 1 and 4

2.7. **Splitting dataset**: Use the function you wrote for subtask 2.3 and select 700 samples from dataset4. Call this new dataset training_set. Call remaining 300 samples as testing_set. Note you need to modify the function in task 2.3 as it needs to return the remained samples as another dataset! **4 Points**

2.8. **Classification:** Train a decision tree with depth=3 using your training_set. Report its classification accuracy using testing_set. Submit the decision tree. **2 Points**



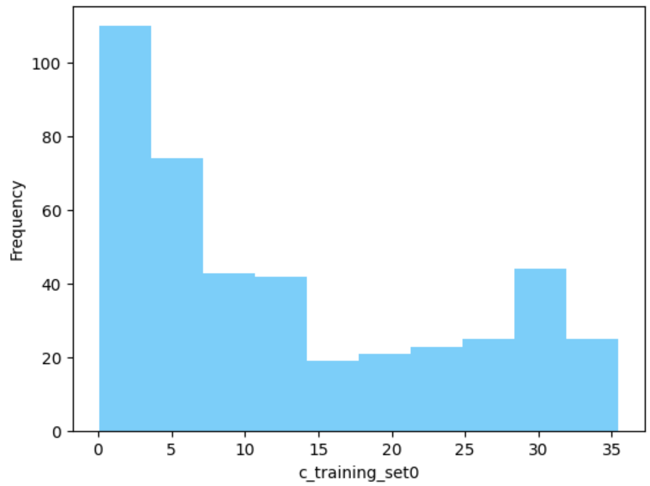
For the classification accuracy, I got 0.88 using testing_set which means that testing_set is good at predicting class.

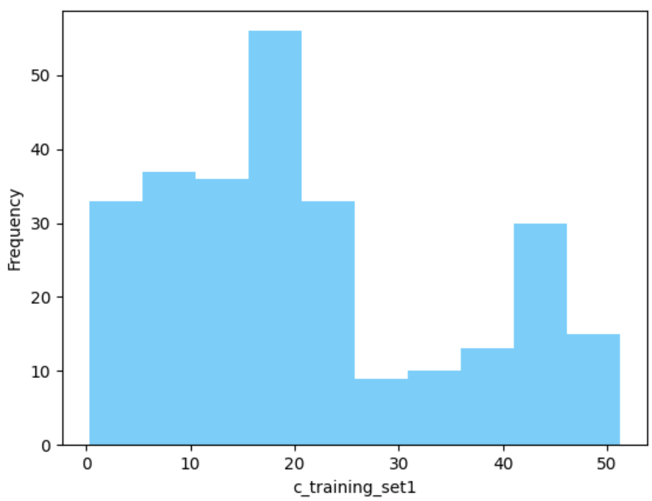
2.9. **Feature creation:** Write a function that accepts a dataset with two features (f_1, f_2) as its input and builds a new dataset with a new feature computed as follows,

$$f_{new} = \sqrt{f_1^2 + f_2^2}$$

Create a new training_set and testing_set by passing training_set and testing_set through this function and call them c_training_set and c_testing_set. **2 Points**

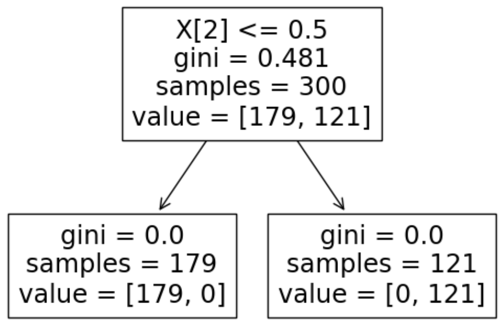
2.10. **Visualization:** Obtain two histograms for c_training_set for the new feature f_{new}, one for the instances of class 0 and one for instances of class 1. Remember to use your personalized colors for two classes! Compare your obtained results with part 2.6 and explain the reasons. **3 Points**





Fnew is calculated in terms of magnitude between feature 1 and 4 so with the new training_set0 that would mean that the overall magnitude shows a certain trend as training_set0 increases or decreases which we can see with the first graph showing a relatively stable downward trend, symbolizing that it would be more useful in predicting than the other histogram for training_set0. For c_training_set 1 the overall magnitude shows a bit more of an erratic trend but still showing a certain downward trend. In comparison with the histograms of 2.6 where most graphs are erratic with little correlation, these graphs tend to have a better relationship with class and so are better used to predict it.

2.11. **Classification:** Train a decision tree with depth=3 using your c_training_set. Report its classification accuracy using c_testing_set. Compare your result with subtask 2.8 and explain the reason. **4 Points**



The classification accuracy I got using the c_testing_set was 0.95, which means that I can infer from the total that the model predicts Class 95% of the time while the model in 2.8 predicts it 0.88 therefore the previous model is underestimating this class. It could be the case that it learned specific rules on the train set, that work against the model in the test set. This means that this model is better at predicting Class with c_testing_set, and worse at predicting with just testing_set.

2.12. **Building a pipeline:** Write a function that accepts

- A dataset
- A variable specifying the sampling method (this variable can be set as “rgl” or “stf” by the user)
- A variable for number of samples in sampled dataset.
- A variable specifying the number of samples in training set

as its input and outputs the classification accuracy. (Call the functions written in previous subtasks in this function. The output of one function must be fed the next one as its input.)

(Take the dataset and all required variables → Sampling based on the selected method → Feature selection → Feature creation → Splitting new dataset to train and test → Train a decision tree with depth=3 → classification accuracy)

Report the classification accuracy for following settings:

1) Main dataset, “stf”, 500, 300
2) Main dataset, “rgl”, 100,70
3) Main dataset, “stf”, 1500,1000

Also, discuss the advantages of using a pipeline. **4 Points**

The importance and usefulness of using a pipeline is that you can easily streamline inputs to smoothly use all the functions that we used so far to get a quick and easy output that is automated to give you the result that you need.

2.13. **Write a conclusion** (at most 20 sentences!) about what you learned in this task and the problems you encountered during writing your code! **5 Points**

Through this assignment I was able to learn a lot more about using python as a programming language to aid in calculating statistical data. From problems 2.2 and 2.3, I was able to learn more about sampling both regular and stratified and how to extract that information using python. From problems 2.4, 2.5, 2.6, and 2.10, I relearned about covariance, its meaning, and plotting both scatter plot and bar graphs to identify any attributes. Problems 2.8 and 2.11, both helped me understand how to create a decision tree and also read the information on it and compare it with each other. Problem 2.9 had me learn more about python math and appending columns to a dataset. Problem 2.11 taught me the importance and usefulness of using a pipeline to streamline and smoothly use all the functions that we used so far to get a quick and easy answer. One of the main problems I encountered during writing my code is trying to find out how rstudio can be converted into python and how to achieve what I did with one of the others using functions that I wasn't familiar with. It took quite some time getting used to the switch from rstudio to python, but I was able to learn how to do so with enough errors and effort I was able to complete all the functions. All these tasks definitely felt like an extension of the first task and reinforced some of what I learned from the first one as well as helping me learn more about manipulating data and dataset with another software. I feel like I'm getting more comfortable with manipulating datasets, sampling, splitting data, creating types of data visualization, classifying and sorting data to generate useful and more understandable ways of analyzing great quantities of data

Remark: *Select your features carefully as the next steps depend on your selection! Most of the points of the tasks will be given to explaining the reasons behind the results. Therefore, by showing only graphs without any discussions you will get a few points of that task! Your report must be at least* ***10 pages long****. Avoid using too large plots in your report!*

How to submit:

Please upload a ***ZIP*** *file including your answers in* ***PDF*** *format and your code files in Blackboard. Your PDF report must contain your explanations and any graphs you plotted.*

```python
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.
from openpyxl import Workbook
import pandas as pd
import numpy as np
import math
from scipy import stats
from matplotlib import pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn import tree


def plot_colors(student_id):
    color1 = "#" + student_id[1:]
    color2 = "#" + str(hex(int("FFFFFF", 16) - int(student_id[1:], 16)))[2:]
    return color1, color2


psid = "1833106"
color1, color2 = plot_colors(psid)
```

```
def newfeature(dataset):
    dataset['fnew'] = dataset.iloc[:, 0].pow(2) + dataset.iloc[:, 0].pow(2)
    dataset['fnew'] = dataset['fnew'] ** (1 / 2)
    return dataset


def myfunc(x):
    return slope * x + intercept


def pipeline(dataset, sampling, numsamples, setnumber):
    return dataset


def regsample(q):
    chosen_idx = np.random.choice(8400, q, replace=False)
    dataset2 = df.iloc[chosen_idx]
    return dataset2


def strasample(q):
    chosen_idx = np.random.choice(8400, q, replace=False)
    dataset2 = df.iloc[chosen_idx]
    return dataset2



# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    df = pd.read_excel(
        r'E:\sources\ExcelTask2.xlsx')  # place "r" before the path string to address special character, such as '\'. Don't forget to put the file name at the end of the path + '.xlsx'
    rows = len(df.axes[0])
    col = df.axes[1]
    # PROBLEM 2.1
    class0 = df.loc[df['Label'] == 0]
    class1 = df.loc[df['Label'] == 1]
```

```
    # print (len(class0))
    # print (len(class1))
    q = 1000
    # PROBLEM 2.2
    dataset2 = regsample(q)
    df20 = dataset2.loc[dataset2['Label'] == 0]
    df21 = dataset2.loc[dataset2['Label'] == 1]
    # print (len(df20))
    # print (len(df21))
    # PROBLEM 2.3
    dataset3 = df
    dataset3 = dataset3.groupby('Label', group_keys=False).apply(lambda x:
x.sample(frac=0.11904761904))
    df30 = dataset3.loc[dataset3['Label'] == 0]
    df31 = dataset3.loc[dataset3['Label'] == 1]
    # print (len(df30))
    # print (len(df31))
    # PROBLEM 2.4
    covMatrix = pd.DataFrame.cov(dataset3)
    # print (covMatrix)
    dataset4 = dataset3
    dataset4 = dataset4.filter(['Feature 1', 'Feature 4'], axis=1)
    # PROBLEM 2.5
    slope, intercept, r, p, std_err = stats.linregress(dataset4['Feature 1'],
dataset4['Feature 4'])
    mymodel = list(map(myfunc, dataset4['Feature 1']))
    plt.scatter(dataset4['Feature 1'], dataset4['Feature 4'], color=color2,
alpha=0.3)
    plt.plot(dataset4['Feature 1'], mymodel)
    plt.show()


    # PROBLEM 2.6
    h0dataset=dataset3.loc[dataset3['Label'] == 0]
```

```python
h1dataset=dataset3.loc[dataset3['Label'] == 1]
h1=h0dataset['Feature 1']
h2=h0dataset['Feature 4']
h3=h1dataset['Feature 1']
h4=h1dataset['Feature 4']

plt.xlabel("Class0 Feature 1")
plt.ylabel("Frequency")
plt.hist(h1, color=color2)
plt.show()

plt.xlabel("Class0 Feature 4")
plt.ylabel("Frequency")
plt.hist(h2, color=color2)
plt.show()

plt.xlabel("Class1 Feature 1")
plt.ylabel("Frequency")
plt.hist(h3, color=color2)
plt.show()

plt.xlabel("Class1 Feature 4")
plt.ylabel("Frequency")
plt.hist(h4, color=color2)
plt.show()

# Problem 2.7
training_set = dataset3
training_set = training_set.filter(['Feature 1', 'Feature 4', 'Label'], axis=1)
training_set = training_set.sample(n=700)
datasetlabel = dataset3.filter(['Feature 1', 'Feature 4', 'Label'], axis=1)
```

```
                testing_set     =     pd.concat([datasetlabel,     training_set,
training_set]).drop_duplicates(keep=False)

   # Problem 2.8

   tree = DecisionTreeClassifier(max_depth=3)

   tree.fit(training_set, training_set['Label'])

   tree_dot = plot_tree(tree)

   plt.show()

   # Problem 2.9

   c_training_set = newfeature(training_set)

   c_testing_set = newfeature(testing_set)

   # Problem 2.10

   c_training_set0 = c_training_set.loc[c_training_set['Label'] == 0]

   c_training_set1 = c_training_set.loc[c_training_set['Label'] == 1]

   plt.xlabel("c_training_set0")

   plt.ylabel("Frequency")

   plt.hist(c_training_set0['fnew'], color=color2)

   plt.show()


   plt.xlabel("c_training_set1")

   plt.ylabel("Frequency")

   plt.hist(c_training_set1['fnew'], color=color2)

   plt.show()

   # Problem 2.11

   tree = DecisionTreeClassifier(max_depth=3)

   tree.fit(training_set, training_set['Label'])

   tree_dot = plot_tree(tree)

   plt.show()

   # Problem 2.12

   pipeline(df, 'stf',  500, 300)

   pipeline(df,  'rgl', 100,70)

   pipeline(df, 'stf', 1500,1000)
```