Homework 1

Syed Faizan Ali (UFID : 26592828)

Charles Richardson (UFID : 73112398)

```
# import required libraries
import numpy as np
from numpy.linalg import norm
import pandas as pd
import math
```

Q1: Attribute Classification

Binary, Discrete, Continuous

Qualitative (Nominal/Ordinal), Quantitative (Interval/Ratio)

1. Discrete Qualitative Ordinal.

   > Although it is a number, it doesn't make sense to +,-,*,/ house number’s

2. Continuous Quantitative Ratio
3. Discrete Qualitative Nominal
4. Discrete Qualitative Nominal

   > Quantitative Interval if numbers ordered

5. Continuous Quantitative Ratio
6. Continuous Quantitative Ratio
7. Discrete Qualitative Nominal
8. Discrete Quantitative Interval
9. Discrete Qualitative Nominal

   > Although it is numeric, it doesn't make sense to +, -, *, / Student ID’s

10. Discrete Quantitative Interval

Q2: Customer Transaction Database

1. From the perspective of the business binary attributes are **asymmetric**, because a purchased item is a much more significant state than not purchasing a product.
2. Jaccard Coeff better reflects the true customer similarity because it portrays that the customers are less similar than the Simple Matching Coeff, giving a better indication of similarities.
   - SMC(1, 2) = 0.9
   - SMC(1, 3) = 0.7
   - JC(1,2) = 0.67
   - JC(1,3) = 0.0
3. Cosine Similarity is the best fit for this application because of its strength with high-dimensional data and its consideration for the importance of the individual items. Euclidean distance is sensitive in high dimensional datasets, SMC/JC are for binary vectors, and correlation is affected by outliers.

Q3: p & q

1. See code output
2. See code output. When q is adjusted by +3, resulting in 0's through out the vector, Cosine correlation changes to match Pearsons Correlation, a perfect direct correlation
3. See code output. When the absolute magnitude of q is multiplied, Cosine correlation gets slightly worse, even though the relative codependence of the values is unchanged. Pearsons Correlation does not change.

```
def cosine(A, B):
  return np.dot(A,B)/(norm(A)*norm(B))

p = np.array([1, 0, 1, 0, 1, 0])
q = np.array([3, -3, 3, -3, 3, -3])
print("".join("=" for _ in range(40)))
print("PART 1")
print(f"\tCosine(p,q) = {cosine(p,q):.2f}")
print(f"\tCorrelation(p,q) = {np.corrcoef(p, q)[0,1]:.2f}")

print("\nPART 2")
q1 = [x+3 for x in q]
print(f"\tCosine(p,q) = {cosine(p,q1):.2f}")
print(f"\tCorrelation(p,q) = {np.corrcoef(p, q1)[0,1]:.2f}")

print("\nPART 3")
q2 = [x*3 for x in q]
```

```
print(f"\tCosine(p,q) = {cosine(p,q2):.2f}")
print(f"\tCorrelation(p,q) = {np.corrcoef(p, q2)[0,1]:.2f}")
print("".join("=" for _ in range(40)))
```

```
========================================
PART 1
        Cosine(p,q) = 0.71
        Correlation(p,q) = 1.00

PART 2
        Cosine(p,q) = 1.00
        Correlation(p,q) = 1.00

PART 3
        Cosine(p,q) = 0.71
        Correlation(p,q) = 1.00
========================================
```

Q4: Binary Classification Problem

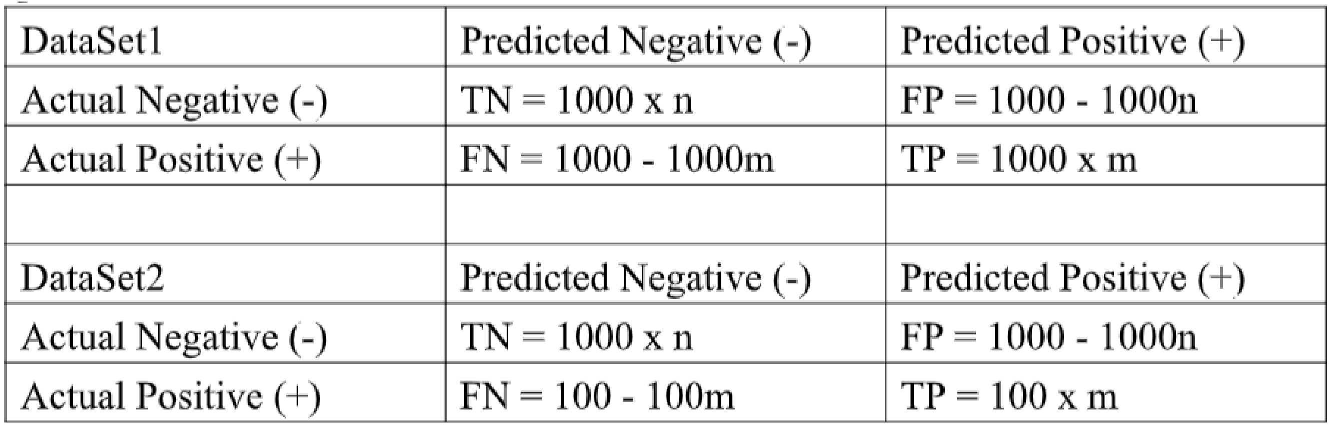
1. See code output
2. See code output
3. See code output. The Customer ID provides the highest information gain. Gender and Marital Status provide the lowest information gains of 0.
4. For splitting at the root node, we choose the attribute with the highest infomation gain, hence Customer ID
5. Extra Credit
    - Tree 1
        - Weighted entropy of the leaves is equal to the calculation in 3. 1 - 0 = 1
    - Tree 2
        - E = 0. Therefore, decrease 1 - 0 = 1
    - Tree 3
        - E = 1.83. Therefore, decrease 1 - 1.83 = -0.83

    1. Tree 3 would be chosen for classification. No it is not the same roots chosen in D.
    2. Entropy was used as the impurity measure, using it to calculate Information Gain of an entropy split. Entropy is a measure of disorder, and information gain is a measure of clarity following a split. The attribute with the highest information gain must always be chosen.

```
data = {'Customer ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16],
        'Housing Type': ['Apartment', 'House', 'House', 'Apartment', 'Apartment', 'Ho
        'Gender': ['Male', 'Male', 'Female', 'Female', 'Male', 'Male', 'Female', 'Fem
        'Marital Status': ['Married', 'Single', 'Married', 'Single', 'Married', 'Sing
        'Class': ['C0', 'C1', 'C1', 'C0', 'C0', 'C1', 'C1', 'C0', 'C0', 'C1', 'C1', '

df = pd.DataFrame(data)
output_part_b = pd.DataFrame(columns=['Attribute', 'Value', 'Entropy'])
output_part_c = pd.DataFrame(columns=['Attribute', 'Information Gain'])


class_counts = df["Class"].value_counts()
class_probs = class_counts / len(df)
entropy = -sum([p * math.log2(p) for p in class_probs])

for col in df.columns[:-1]:
    col_entropy = 0
    col_counts = df[col].value_counts()
    for value, count in col_counts.items():
        value_data = df[df[col] == value]
        value_class_counts = value_data['Class'].value_counts()
        value_class_probs = value_class_counts / len(value_data)
        value_class_entropy = -np.sum(value_class_probs * np.log2(value_class_probs))
        output_part_b = output_part_b.append({'Attribute': col, 'Value': value, 'Entr
        col_entropy += (count / len(df)) * value_class_entropy
    ig = entropy - col_entropy
    output_part_c = output_part_c.append({'Attribute': col, 'Information Gain': ig.ro

print("".join("=" for _ in range(40)))
print("PART A (1) ")
print(f"The entropy of the class distribution is {entropy:.2f}")

print("\nPART B (2)")
print(output_part_b.to_string(index=False))

print("\nPART C (3)")
print(output_part_c.to_string(index=False))
print("".join("=" for _ in range(40)))
```

```
========================================
PART A (1)
The entropy of the class distribution is 1.00

PART B (2)
     Attribute     Value  Entropy
   Customer ID         1    -0.00
   Customer ID         2    -0.00
   Customer ID         3    -0.00
   Customer ID         4    -0.00
   Customer ID         5    -0.00
   Customer ID         6    -0.00
   Customer ID          7    -0.00
   Customer ID          8    -0.00
   Customer ID          9    -0.00
   Customer ID         10    -0.00
   Customer ID         11    -0.00
   Customer ID         12    -0.00
   Customer ID         13    -0.00
   Customer ID         14    -0.00
   Customer ID         15    -0.00
   Customer ID         16    -0.00
  Housing Type Apartment    -0.00
  Housing Type     House     0.92
  Housing Type    Hostel    -0.00
        Gender      Male     1.00
        Gender    Female     1.00
Marital Status   Married     1.00
Marital Status    Single     1.00

PART C (3)
     Attribute  Information Gain
   Customer ID              1.00
  Housing Type              0.66
        Gender              0.00
Marital Status              0.00
========================================
```

Q5

1. 

DataSet1	Predicted Negative (-)	Predicted Positive (+)
Actual Negative (-)	TN = 1000 x n	FP = 1000 - 1000n
Actual Positive (+)	FN = 1000 - 1000m	TP = 1000 x m
DataSet2	Predicted Negative (-)	Predicted Positive (+)
Actual Negative (-)	TN = 1000 x n	FP = 1000 - 1000n
Actual Positive (+)	FN = 100 - 100m	TP = 100 x m

2.

- Dataset 1
  - (TP + TN) / (TP + TN + FP + FN) = 1000n + 100m / 2000
  - Precision = TP / (TP + FP) = m / m + 1 - n
  - TPR = TP / (TP + FN) = m
  - FPR = FP / (FP + TN) = 1-n
- Dataset 2

- (TP + TN) / (TP + TN + FP + FN) = 100m + 1000n / 1100
- Precision = TP / (TP + FP) = m / m + 10 - 10n
- TPR = TP / (TP + FN) = m
- FPR = FP / (FP + TN) = 1-n

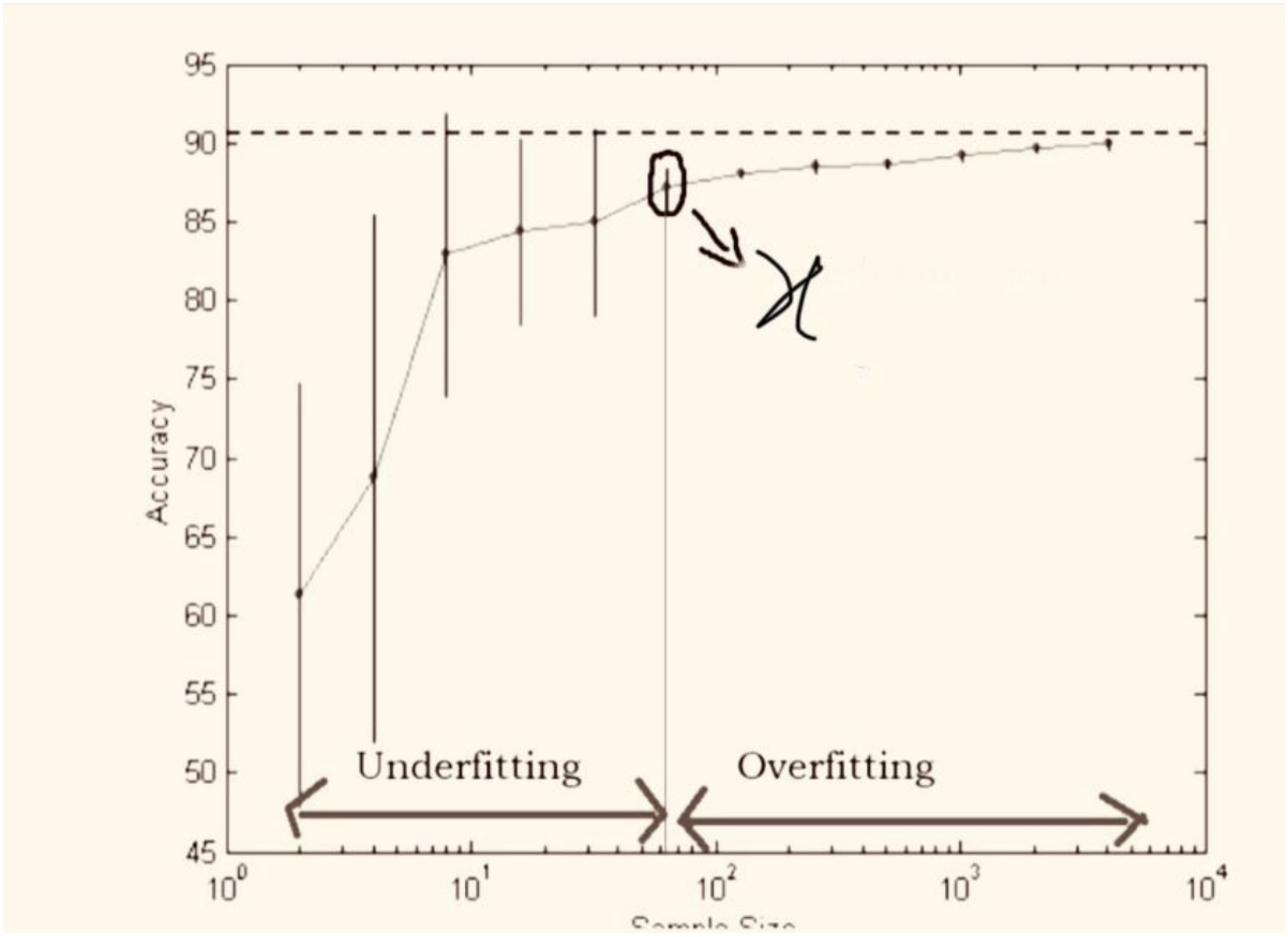
3. - If the skew in the test data is 1:s, then the accuracy of hte algorithm on this data set is
     - Accuracy = (TP + TN) / (TP + TN + FP + FN) = no of positive is 1, no of negative is s = (m + (s*n) / (1 + s))
     - s = large
       - accuracy = m + s.n / 1 + s = s * n / s = n
     - s = small
       - accuracy = m + s.n / 1 + s = m * s.n = m
   - If s is very large (>>1), then the test data will have a significant skew toward the negative class. In this case, the accuracy of the algorithm will be very close to n, since hte classicier is likely to predict most instances as negative. If s is very small (<<1) then the test data is skeward towards the positive class. In this case the accuracy of the algorithm will be close to m, since the classifier is likely to predict most instances as positive.
4. In scenarios where class imbalance is high, precision and recall are better metrics than overall accuracy. Precision captures the fraction of the instances that are truly positive among all instances that are predicted as positive. It is a measure of how confident we are about the predicted positive instances. On the other hand, recall captures the fraction of the truly positive instances that are correctly predicted as positive. It is a measure of how well the classifier can identify positive instances. Overall accuracy can be misleading in such scenarios.
   - S = large
   - Accuracy = n
   - Precision = TP / TP + FP = m / m + (1 - n)*s = m / (m + s - ns) = m / m + s = m / s = 0
   - Recall = TP / TP + FN = m / m + (1-m) * 1 = m / m + 1 - m = m
   - While Accuracy does not provide this information, Precision and Recall together provide all the parameter information.
   - While recall does not provide us with that information, if precision is 0, it informs us that the dataset is unbalanced.

Q6

1. Decision Trees is the best fit given its ability to handle and filter irrelevant attributes. KNN' and ANN's dependencies on distance metrics or NN connections, respectively could make them sensitive to irrelevant attributes.

2. Decision Trees is also the best fit in this scenario, as it selectively handles attributes to create complex decision rules. Naive Bayes is feasible as well due to its interrelating properties of field characteristics. The chance KNN will not perform will is high because of its dependence on distance measurements, which will result in shortcomings in the insights of interactions among features.

Q7



- Accuracy: While a model has a lot to learn, when it is underfitting, its accuracy levels remain low. As sample numbers rise, the relative accuracy increases tend to decline as the model becomes overfit because there isn't much new data for it to take into account.

- Standard Deviation: The accuracy levels in different trails tend to be closely spaced when a model is overfitting since the model has learned nearly everything from the sample points.