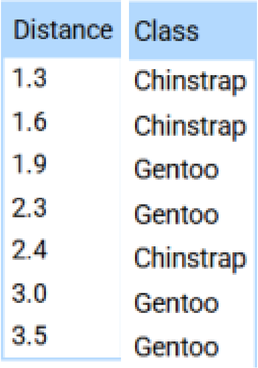
Q1:KNN

Based on the given table, we are given distances, bill length, bill depth, and
Class. For finding KNN we follow 3 steps which is 1.Measure the distance,Find the K nearest neighbors,and majority voting and for this problem, we need to classify the data in the table based on k=1,3,5. For starters, step one is already mostly done because we are given the distance so we do not need to solve for it. That means we do not need to worry about the extra information about bill length and depth. We can now reduce the relevant information on the table to this:



Distance	Class
1.3	Chinstrap
1.6	Chinstrap
1.9	Gentoo
2.3	Gentoo
2.4	Chinstrap
3.0	Gentoo
3.5	Gentoo

For k=1:

Nearest neighbor is 1.3 distance because that is the first neighbor that is closest to our root, and this class is Chinstrap.Since this is the only k(point) we had to consider, our prediction for when k=1, our prediction will be Chinstrap

For k=3:

The 3 closest k neighbors are as follows:Chinastrap=1.3,Chinstrap=1.6, and Gentoo = 1.9
Since these 3 have the smallest distance values relative to root. Since for this k, we have 2 Chinstrap, our prediction for the next point will be Chinstrap because it has the majority vote with 2k.

For k=5:

Our nearest 5 distances are Chinstrap=1.3,Chinstrap=1.6, Gentoo=1.9,Gentoo=2.3, and Chinstrap=2.4. In comparison, there is 3 Chinstrap vs 2Gentoo, so since Chinstrap>Gentoo, our predicted point will be Chinstrap again.

Q2.

```
import matplotlib.pyplot as plt
import numpy as np
m=1# degree of the polynomial regression function
M={1,3,5,10,15}
x = np.array([52,55,50,70,75,72,80,82,85,97,95,90,105,108,100])
y = np.array([3.3,2.8,2.9,2.3,2.6,2.1,2.5,2.9,2.4,3.0,3.1,2.8,3.3,3.5,3.0])
mymodel = np.poly1d(np.polyfit(x, y, 1))
plt.scatter(x, y)
myline = np.linspace(40, 110, 100)
```

```
plt.plot(myline, mymodel(myline))

plt.show()
print(mymodel.coefficients)
```

Calculate the cost for training dataset and also for the validation set with learned regression models. Use the cost equation that we learned in the class.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

M={1,3,5,10,15}

When M = 3
y^=(a3)x^3+(a2)x^2+(a1)x+a0
For training
=1/10[(3.3−3.01)^2+(2.9−2.97)^2+(2.3−2.36)^2+(2.1−2.22)^2+(2.5−2.48)^2+(2.4−2.55)^2+(3.0−2.99)^2+(2.8−2.75)^2+(3.5−3.47)^2+(3.0−3.02)^2]
=0.1364/10=0.0246
For validation: 1/5[(2.8−2.87)^2+(2.6−2.31)^2+(2.9−2.52)^2+(3.1−2.88)^2+(3.3−3.42)^2]
1/5[0.0049+0.0841+0.1444+0.0484+0.0144]
= 0.2962/5 = 0.0759
…
M = 1 training = 0.1523 validation = 0.1099
M = 5 training = 0.0075 validation = 0.1861
M = 10 training = 0 validation = 24.44
M = 15 training = 0 validation = 73

e. Decide which model is better to select and explain your answers.
Best model is when degree is 3 or 5 because it has low training error and decent validation error as when M=1, there is high training and validation error, and when M = 10,15 there is 0 training error but high validation error so 3 or 5 is the best model since it has the best balance.
Q3

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn import metrics

# 1. Load and Prepare the Iris Dataset
iris = load_iris()
```

```
X_iris = iris.data
y_iris = iris.target

# training and testing
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_iris, y_iris, test_size=0.3,
random_state=42)

clf_iris = DecisionTreeClassifier(max_depth=5)  # Limit depth to 5 to avoid a too deep tree
clf_iris.fit(X_train_iris, y_train_iris)

# Evaluate the test
y_pred_iris = clf_iris.predict(X_test_iris)
accuracy_iris = metrics.accuracy_score(y_test_iris, y_pred_iris)
print(f"Accuracy on Iris dataset: {accuracy_iris * 100:.2f}%")

# Plot the decision tree
plt.figure(figsize=(15,10))
plot_tree(clf_iris, filled=True, feature_names=iris.feature_names,
class_names=iris.target_names, rounded=True)
plt.title("Decision Tree for Iris Dataset")
plt.show()


col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
pima = pd.read_csv("diabetes.csv", header=None, names=col_names)

X_pima = pima.drop('label', axis=1)
y_pima = pima['label']

# training and testing datasets
X_train_pima, X_test_pima, y_train_pima, y_test_pima = train_test_split(X_pima, y_pima,
test_size=0.3, random_state=42)

# Decision Treemodel
clf_pima = DecisionTreeClassifier(max_depth=5)  # Limit depth to 5 to avoid a too deep tree
clf_pima.fit(X_train_pima, y_train_pima)

# Evaluate
y_pred_pima = clf_pima.predict(X_test_pima)
accuracy_pima = metrics.accuracy_score(y_test_pima, y_pred_pima)
print(f"Accuracy on Diabetes dataset: {accuracy_pima * 100:.2f}%")

# Plot the decision tree
plt.figure(figsize=(15,10))
```

```
plot_tree(clf_pima, filled=True, feature_names=X_pima.columns, class_names=["Negative",
"Positive"], rounded=True)
plt.title("Decision Tree")
plt.show()
```

Q4

1.

P(B|A)=P(A|B)P(B)/ P(A)

Stolen= 5, Not Stolen= 5 total cars = 10

Stolen

P(Stolen) = 5/10 = 0.5, P(Not Stolen) 5/10=0.5

From the given table, P(red |Stolen) = ⅗ = 0.6

P(Suv | Stolen) = ⅕ = 0.2

P(Domestic | Stolen) ⅖ = 0.4

P(A|Stolen)=0.6×0.2×0.4=0.048

P(Stolen|A)=0.048×0.5=0.024

Not Stolen

P(Red | Not Stolen = ⅖ = 0.4

P(SUV | Not Stolen) = ⅖ = 0.4

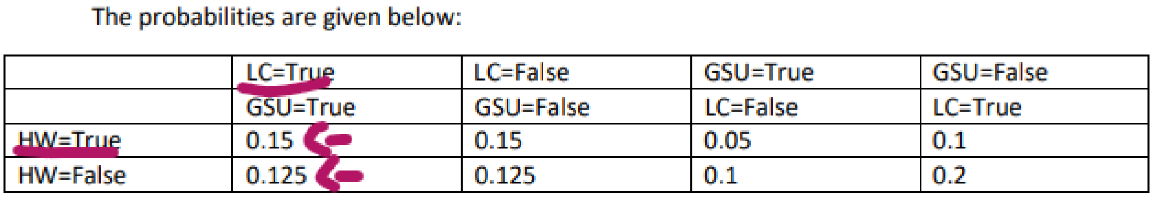
P(Domestic | Not Stolen) = ⅖ = 0.4

P(A|Not Stolen)=0.4×0.4×0.4=0.064

P(Not Stolen|A)=0.064×0.5=0.032

Since our probabilities on P(Stolen|A)==0.024 < P(Not Stolen|A)=0.032, we can predict that the (Red, SUV, Domestic) car is most likely not stolen

2



The probabilities are given below:

	LC=True	LC=False	GSU=True	GSU=False
	GSU=True	GSU=False	LC=False	LC=True
HW=True	0.15	0.15	0.05	0.1
HW=False	0.125	0.125	0.1	0.2

a)

P(HW=T,LC=True, GSU=True)=0.15

First we are doing HW=True LC=True + HW=False LC=True

0.15 + 0.125 = 0.275

Then we calculate for LC =False and LC=True where both HW is True

0.15+0.15 = 0.3

Now the reason why we are getting these values is because they identify as our event A (values solved from given table) and B(value given to us) to test for independence as from probability

and statistics, P(A n B) = P(A)P(B) if they are independent and we are comparing these two values to solve for independence.
So for when LC=True and HW=True, it was given to be 0.15.
P(LC=True)P(Hw=True) = P(HW=True)P(LC=False)
0.15 = (.275)(0.3)
0.15 = .0825
Since they are not equal, “Person X likes coding” is not independent of “person X is hard-working” because it fails P(A n B) = P(A)P(B)

b)
Our condition we are solving for is P(LC|HW,GSU) = P(LC|GSU)
P(B|A) = P(A)P(B)/P(A)
Where P(B|A) = P(LC=True∣HW=True,GSU=True)
P(A)=LC=True
P(B)=P(HW=True, GSU+True)
P(HW=True,GSU=True)=P(HW=True,LC=True,GSU=True)+P(HW=True,LC=False,GSU=True)
=0.15+0.15=0.30
P(LC=True∣HW=True,GSU=True)=
0.15/.30 =0.5
P(LC=True∣GSU=True) = P(HW=True,LC=True,GSU=True)+P(HW=False,LC=True,GSU=True)
0.15 + 0.125 = 0..275
(GSU=True)=P(HW=True,GSU=True)+P(HW=False,GSU=True)
=0.30+0.225=0.525
P(LC=True∣GSU=True)= 0.275/0.525 =0.5238
C is NOT conditionally independent of HW given GSU Since Since
P(LC=True∣HW=True,GSU=True)≠P(LC=True∣GSU=True)P(LC=True | HW=True, GSU=True)
P(LC=True | GSU=True)P(LC=True∣HW=True,GSU=True) does not equal to
P(LC=True∣GSU=True)

c)
What we are finding:P(HW=True∣GSU=True)
P(HW=True∣GSU=True)= P(GSU=True), P(HW=True/GSU=True)
From previous questions, we found P(HW=True,GSU=True)=0.30, P(GSU=True)=0.525
P(HW=True∣GSU=True)= 0.3/0.525=0.5714
So 57.14% chance that the person attends GSU and is hard working