A. Mathematics:

1. In class, we discussed the multinomial likelihood function: \( P(x|\Theta) = \prod_{j\in Y} (p_y^j)^{\text{Count}_j} \)
   where \( p_y^j \) is the probability event \( j \) happens in class \( y \) and \( \text{Count}_j \) is the number of times this event occurs in the data \( x \). For example, we may wish to classify a cat as angry, happy, sad, or tired (4 potential classes \( y \)) based on an observed sequence of cat activities: eating, hiding, napping, pouncing, or stretching (5 potential activities). An input sequence \( x \) may be: nap, nap, stretch, nap, eat.

We wish to estimate the parameters \( \Theta = \{p_y^j\} \forall y, j \) given a set of training data \( \{X, y\} - 1000 \) distinct sequences of events and the corresponding class labels for each sequence.

(a) To find the MLE value of all \( p_y^j \) for a given class \( y \), you must solve simultaneously for all values of \( j \), while also enforcing \( \sum_j p_y^j = 1 \), i.e., by expressing the probability of the last possible event/activity as \( p_{\text{last}} = 1 - \sum_{j\neq\text{last}} p^i \). For example, in the cat mood problem, we would solve:

\[
\frac{d}{dp_{\text{sad}}} P(x|\Theta) = 0, \quad \frac{d}{dp_{\text{eat}}} P(x|\Theta) = 0, \quad \frac{d}{dp_{\text{nap}}} P(x|\Theta) = 0, \quad \frac{d}{dp_{\text{pounce}}} P(x|\Theta) = 0
\]

and \( p_{\text{sad}}^{\text{stretch}} \) would be \( 1 - (p_{\text{sad}}^{\text{eat}} + \cdots + p_{\text{sad}}^{\text{pounce}}) \).

Provide at least 3 mathematical steps to derive the MLE estimate of \( p_y^j \), presuming more than 2 possible values for event \( j \). (You do not have to provide the final answer, only 3 steps towards the answer are required.)

(b) Let us now presume there are only two possible events (like the binary coin flip example), and thus there is only one parameter to estimate per class: \( p_y \). Using a Beta prior, we know the MAP estimate would be \( \frac{\#\text{Events} + \alpha - 1}{\#\text{Events} + \alpha + \beta - 1} \).

Let us now presume the prior instead is \( P(p_y) = \alpha (p_y - 0.5)^2 \), what is/are the MAP value(s) of \( p_y \)?

(You may need to recall the roots of the quadratic function \( ax^2 + bx + c \) are: \( \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \))
2. Presuming a super-Gaussian generalized normal likelihood:
\[
P(x|\mu, \sigma) = \frac{4}{2\sqrt{2}\sigma\Gamma(0.25)} \exp \left( -\frac{(x-\mu)^4}{2\sigma^4} \right)
\]
and assuming 1000 training data points used to compute a likelihood: \( L(x; \mu, \sigma) = \prod_i P(x_i|\mu, \sigma) \), solve for the MLE value of \( \sigma \). Show at least two arithmetic steps towards your derivation.

3. Compute the real value(s) of \( \tau \) at which this function reach a slope of 0, and state whether it reaches a maximum or minimum at each point.
\[
f(\tau) = \sum_{l=1}^{20} \frac{6\tau^4}{12l}
\]

4. Consider a two-dimensional generalized Normal distribution \( P(x|\mu, \sigma) \) defined as:
\[
P(x|\mu, \sigma) = N(x_1|\mu_1\sigma_1, 2)N(x_2|\mu_2\sigma_2, 2)
\]
where
\[
N(x|\mu, \sigma, q) = \frac{q}{2\sqrt{2}\sigma\Gamma(1/q)} \exp \left( -\left( \frac{|x-\mu|}{\sqrt{2}\sigma} \right)^q \right)
\]
(for example, when \( q=2, \Gamma(q) = \sqrt{\pi} \) and \( N(x|\mu, \sigma, 2) \) is a regular one-dimensional Gaussian distribution.

   a. Determine the values for \( \mu, \sigma, q \) so that the two dimensional distribution \( P(x|\mu, \sigma) \) will form an **oval** isocontour \( P(x|\mu, \sigma) = 0.5 = k \) as \( m = 0.5 = 2(x_1 - 2)^2 + 0.3(x_2 + 1)^2 \) as shown in blue above, where \( k \) and \( m \) are constants. You must solve for \( \mu \) and \( q \) are required. A correct solution for \( \sigma \) will get you a small amount of extra credit.

   b. What parameters \( (\mu, \sigma, q) \) should we adjust to form a diamond-like isocontour
   \( m = 0.5 = 2|x_1 - 2| + 0.3|x_2 + 1| \), as shown in red, where \( m \) is a constant? You must address \( \mu \) and \( q \). A correct answer for \( \sigma \) will get you a small amount of extra credit.

5. A “positive definite” matrix is any matrix \( A \) such that every vector \( x \) yields a matrix product:
\[
x^TAx \geq 0.
\]
Given the matrix \( A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \), find a vector \( x \) that produces a **negative** output \( x^T Ax \).
6. I have a parrot named Polly. She says “hello” for many reasons. If my wife is entering our house, there is a 40% chance Polly will say “hello.” At any given moment, there is a 30% chance Polly is saying hello and a 10% chance my wife is entering the house.

   a) What is the probability that my wife is entering our house and Polly is not saying Hello?

   b) Right now, I hear Polly saying hello. What is the probability my wife is not entering our house?

7. You have developed a program that determines from a user’s movie-watching history whether s/he is an adult or a teenager. We know 15% of users are teenagers. If user X is a teenager, the program will say so with probability 90%. If user X is an adult, the program will say s/he is an adult with probability 20%.

   Assume that the program says user X is a teenager. What is the probability s/he is actually an adult?

8. What $\alpha$, $\beta$ combination will produce each of the Beta distribution likelihoods below. (All but one option fits a curve)
   - Option I : $\alpha = 5, \beta = 5$
   - Option II : $\alpha = 30, \beta = 9$
   - Option III : $\alpha = 4, \beta = 12$
   - Option IV : $\alpha = 6, \beta = 3$
   - Option V : $\alpha = 10, \beta = 3$

9. What $\alpha$- and $\beta$ value combination for the Beta distribution will approximate (or exactly define) a uniform probability? $P(x) = 1$ for $0 \leq x \leq 1$, $P(x) = 0/\text{undefined otherwise.}$
10. Consider a Bayesian classification problem where we wish to determine if a newly spotted animal is a giraffe. We will use three different features, each capable of taking on a discrete number of values:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Small, Middle-size, Large</td>
</tr>
<tr>
<td>Action</td>
<td>Eating, Walking, Sleeping</td>
</tr>
<tr>
<td>Color</td>
<td>Blue, Green, Orange, Black, White</td>
</tr>
</tbody>
</table>

We classify each animal as either \( y_i = \text{Giraffe} \) or \( y_i = \text{Non-Giraffe} \). Based on a large training set, we wish to estimate all joint probability likelihoods and priors.

a) Assuming the features are not independent, how many total parameters need to be estimated to enable Bayes classification based on the posterior?

b) Using the Naïve Bayes assumption, how many total parameters need to be estimated to enable Bayes classification based on the posterior?

11. I have written a classifier to determine if my dog is sick or healthy. I record the sounds my dog makes once a minute for six minutes, obtaining six sound measurements \( s_1, s_2, \ldots, s_6 \). At each time, there is a likelihood my dog will make the sounds: bark, growl, no-sound, pant, whimper:

| Sound   | \( P(s | y=\text{sick}) \) | \( P(s | y=\text{healthy}) \) |
|---------|------------------|------------------|
| Bark    | 0.2              | 0.4              |
| Growl   | 0.05             | 0.2              |
| No-sound| 0.4              | 0.1              |
| Pant    | 0.1              | 0.15             |

We compute \( P(y | s_1, \ldots, s_6) = P(y) \prod_j P(s_j | y) \)

Note, \( P(y=\text{healthy})=0.8 \)

For each sequence of sounds, provide the \( y=\text{sick} \) and \( y=\text{healthy} \) posterior probability estimates for each class using \( P(y | x) = P(x | y) P(y) \). Also provide the selected class for each sequence using Bayes (posterior) classification.

a) \( s_1 = \text{Growl}, s_2 = \text{Bark}, s_3 = \text{Growl}, s_4 = \text{Whimper}, s_5 = \text{Bark}, s_6 = \text{No-sound} \)

b) \( s_1 = \text{No-sound}, s_2 = \text{Bark}, s_3 = \text{No-sound}, s_4 = \text{Pant}, s_5 = \text{Bark}, s_6 = \text{No-sound} \)

c) \( s_1 = \text{Whimper}, s_2 = \text{Whimper}, s_3 = \text{Bark}, s_4 = \text{No-sound}, s_5 = \text{Bark}, s_6 = \text{No-sound} \)
B. Programming

**Detailed submission instructions:** Code should be left in your private/CIS5800/ directory. Include all function definitions and your answers to question 6 (as a comment) in the file hw1.py.

In class, we discussed classification using the Bayesian Posterior. For this assignment, you will create both classifiers to label online shoppers based on their purchase history. Specifically, you will determine if each shopper is a “Minor”, a “Teen”, or an “Adult”. You will make this determination based on a single feature $x$ – the number of times the shopper has ordered alcohol in the past month.

Assume a Gaussian likelihood $P(\text{amountAlcohol}|\text{shopperAge})$. 

- **amountAlcohol** is a real value corresponding to the number of times the shopper has ordered alcohol in the past month.
- **shopperAge** is a member of the set of classes $Y=\{\text{Minor, Teen, Adult}\}$.

Our first job is to learn the parameters to best describe our likelihood function for each class. Recall the Gaussian is described by mean $\mu$ and variance $\sigma$. For convenience, we will assume a standard deviation of $\sigma = 2$. We can determine the mean by finding the average amount of alcohol for each class, i.e., for each “shopper.”

**Accessing our data**

The file hw1data.npz is available on our website (and on erdos using `cp ~dleeds/MLpublic/hw1data.npz`). Load this file into Python to get access to two data sets `trainData` and `testData`. Each set is a numpy-array. After loading, you should convert the data to Python list format, where each list has dimensions $(2,n)$ containing class labels in the first row/sub-list, e.g., `trainData[0]`, and alcohol quantities in the second row/sub-list, e.g., `trainData[1]`. This loading/conversion can be done as follows:

```python
inDat=np.loadz('MLpublic/hw1data.npz')
trainData=inDat['trainData'].tolist()
trainData[1]=[int(i) for i in trainData [1]]
testData=inDat['testData'].tolist()
testData[1]=[int(i) for i in testData [1]]
```

In addition to the functions specified in the Python intro slides, you may use numpy.loadz to load the data.
Programming assignments:

1. Write a function called `learnMeans` that takes in a Python list with dimensions (2,n) containing class labels in the first list `inputList[0]` and alcohol quantities in the second list `inputList[1]`, and returns MLE estimates of mean alcohol quantities for each class as a dictionary. Use the following syntax for calling `learnMean`:

   ```python
   meansOut=learnMeans(inputList)
   ```

   `meansOut` will be a dictionary with a key for each class: Minor, Teen, and Adult and a real-number value for each key. For example, `inputList` could be:

   ```python
   [['Adult', 'Adult', 'Minor', 'Teen', 'Minor'], [10, 12, 2, 5, 4]]
   ```

   and the `meansOut` would be `{ 'Adult': 11, 'Minor': 3, 'Teen': 5 }`

2. Let us consider MAP learning of Gaussian means. Instead of simply consider the observed alcohol quantities, we also consider a general prior for mean values across all classes:

   \[
   P(\mu; \mu_{\text{gen}}, \sigma_{\text{gen}}) = \frac{1}{\sqrt{2\pi}\sigma_{\text{gen}}} \exp \left( -\frac{(\mu - \mu_{\text{gen}})^2}{2\sigma_{\text{gen}}^2} \right). 
   \]

   (a) What is the MAP formula for the estimate of a single class’s mean, given the prior \( P(\mu; \mu_{\text{gen}}, \sigma_{\text{gen}}) \). Provide this answer as a comment in hw1.py

   (b) Now, write `learnMeanMAP`. Use the following syntax for calling `learnMeanMAP`:

   ```python
   meansOut=learnMeanMAP(inputList, muGen, sigGen)
   ```

   As before, `inputList` is a Python list with dimensions (2,n) containing class labels in the first list `inputList[0]` and alcohol quantities in the second list `inputList[1]`. `muGen` (\( \mu_{\text{gen}} \)) and `sigGen` (\( \sigma_{\text{gen}} \)) control the prior distribution on estimates for all class means. `meansOut` will have a key for each class: Minor, Teen, and Adult and a real-number value for each key.

4. Write a function called `learnPriors` that takes in a Python list with dimensions (2,n) containing class labels in the first list `inputList[0]` and alcohol quantities in the second list `inputList[1]`, and returns prior class probabilities for each of the four shopper classes. Use the following syntax for calling `learnPriors`:

   ```python
   classPrior=learnPriors(inputList)
   ```

   `classPrior` will be a dictionary with a key for each class: Minor, Teen, and Adult and a real-number value for each key – a probability between 0 and 1.

5. Write a function called `labelGB` (for Gaussian Bayes) that takes in an amountAlcohol measurement, a dictionary containing the mean amountAlcohol values for each of the four shopper classes, and a dictionary containing the shopper prior probabilities. `labelGB` then will return the Bayes classifier label for the input amountAlcohol measurement. **The function is to return the class label (Minor, Teen, or Adult) with highest probability.**

   Use the following syntax for calling `labelGB`: 

   ```python
   ```
\[
\text{shopper} = \text{labelGB}(\text{amountAlcohol}, \text{meansDict}, \text{priorsDict})
\]

\text{meansDict} must have the structure of the dictionary output by \text{learnMeans}; \text{priorsDict} must have the structure of the dictionary output by \text{learnPriors}. 

6. Write a function called \textbf{evaluateGB} that takes in all the test data as a Python list with dimensions \((2,n)\) containing class labels in the first list inputList[0] and alcohol quantities in the second list inputList[1], the dictionary of class alcohol means (as specified above), and the dictionary of shopper class priors (as specified above), and outputs the fraction of correctly-labeled data points in the test set. (The fraction will be a decimal number between 0.0 and 1.0, e.g., 0.65)

Use the following syntax for calling \text{evaluateGB}:
\[
\text{evaluateGB}(\text{testData}, \text{meansDict}, \text{priorDict})
\]

We wish to use 5-fold cross-validation to learn and test \(\mu\) values for each shopper class. We consider two methods for cross validation:

\begin{itemize}
  \item \textbf{Ordered}: For loop repetition \(k\), remove data points \((k - 1) \frac{n}{5} + 1 \) to \(k \frac{n}{5} + 1\) for testing, use remaining for training. Save average test set accuracy, average value for each class mean, and average class priors.
  \item \textbf{Stochastic}: Randomly shuffle the order of all data. For example: \[
  \begin{bmatrix}
  \text{'Adult'}, \text{'Minor'}, \text{'Teen'}, \text{'Adult'} \\
  5, 3, 1, 10
  \end{bmatrix}
\] may become \[
  \begin{bmatrix}
  \text{'Minor'}, \text{'Adult'}, \text{'Teen'}, \text{'Adult'} \\
  3, 10, 1, 5
  \end{bmatrix}
\]. After shuffling, repeat the steps from ordered algorithm: For loop repetition \(k\), remove data points \((k - 1) \frac{n}{5} + 1 \) to \(k \frac{n}{5} + 1\) for testing, use remaining for training. Save average test set accuracy, average value for each class mean, and average class priors.
\end{itemize}

Below, use \text{learnMeans} and not \text{learnMeanMAP}

7. Write a function called \textbf{crossValidOrder} that takes in a Python list with dimensions \((2,n)\) as in the functions above, performs ordered cross-validation on the data, and then returns classifier accuracy, learned mean values, and learned class priors.

Use the following syntax for calling \text{crossValidOrder}:
\[
\text{meanAcc, meansDict, priorDict} = \text{crossValidOrder}(\text{dataIn})
\]
8. Write a function called `crossValidStoch` that takes in a Python list with dimensions (2,n) as in the functions above, performs stochastic cross-validation on the data, and then returns classifier accuracy, learned mean values, and learned class priors.

Use the following syntax for calling `crossValidOrder`:
```
meanAcc, meansDict, priorDict = crossValidStoch(dataIn)
```

**For this function (Q8), you also can use the `numpy.random.permutation` function, which returns the input array random shuffled along its first dimension.**

9. If you use the data set called `trainData` for full cross validation (ignoring the set called `testData`), what will be the difference between using `crossValidOrder` and `crossValidStoch`? Provide a 1-2 sentence answer as a comment in `hw1.py`.

Recall, to define a dictionary you can use the command: `myDict={}`
and to add a new key you would use: `myDict[key]=value`

**For your code, you may not import new libraries except those provided in the Python lecture slides. In those imported libraries you can ONLY use commands covered in the Python lecture slides (and numpy.loadz).** (For example, this means you have to implement the Gauss function yourself.)