Andrew Lai
CSC 466: KDD
Prof. Paul Anderson

GAUSSIAN BAYESIAN CLASSIFIER:

The classifier created for this mastery checkpoint had only a few changes from the one seen in lab 2. The primary change was the change in the method to get the probability of any given feature's value. Previously, this probability was looked up in a table of pre-calculated class conditional probabilities, but now is generated using this function:

```python
def getGaussianProbability(x, mean, sd):
    # norm.cdf is probability less than or equal to parameter x
    # uses a fraction of sd to get a range of x values
    b = x + sd*.2
    a = x - sd*.2
    return norm.cdf(b, loc=mean, scale=sd) - norm.cdf(a, loc=mean, scale=sd)
```

Where mean and sd are generated from this function:

```python
# for each feature f:
#       for each target value v:
#           get all entries where target t = v
#           dict[f][v] = (mean, sd)
def getFeatureGaussianModels(X:pd.DataFrame, y:pd.Series):

    featureModels = {}
    for colName, col in X.iteritems():
        targetValues = {}
        for val in y.unique():
            targetValues[val] = (col[y==val].mean(), col[y==val].std())
        featureModels[colName] = targetValues

    return featureModels
```

The post_priors method was then modified to accommodate the gaussian models as such:

```
# x is a series mapping colNames to values
def predictX(gaussianModels:dict, priors:dict, x:pd.Series):

    numerators = {}

    """
    for every value i, v in priors:
        numerator[i] = priors[v]
        for every feature f, val in x:
            multiply numerator[i] by getGaussianLiklihood(gaussianModels[f][v], val)

    denominator = sum(numerators)
    """

    for priorVal, priorProb in priors.items():
        numerator = priorProb
        for colName, val in x.iteritems():
            numerator *= getGaussianProbability(val,
*(gaussianModels[colName][priorVal]))
        numerators[priorVal] = numerator

    denominator = sum(numerators.values())
    post_priors = {}
    max = (0, 0)
    for priorVal, priorProb in priors.items():
        post_priors[priorVal] = numerators[priorVal]/denominator
        if post_priors[priorVal] >= max[1]:
            max = (priorVal, post_priors[priorVal])
    return max
```

The dataset I chose to demonstrate this prediction model on was one from a speed-dating event back in the early 2000's. Each participant would rate their speed-dating partner after a series of 4-minute conversations between them and their partners. At the end of the evening, each participant would say “yes” or “no” to each of the other participants. The researchers also took data on how each participant rated themselves in a series of qualities that people look for in a relationship. I wanted to see if how these participants viewed themselves would be a significant indicator of if others would want to date you or not. In other words, how well do you know yourself and will knowing yourself get you a boyfriend/girlfriend. The results were surprising:

0.5826282161975479

This is the accuracy of the resulting model, just barely better than randomly guessing. With that the following feature importances aren't that surprising:

```
'attractive': 0.0010936510677488354
'sincere': 0.001726817475392865
'intelligence': 0.0010936510677488354
'funny': 0.0005180452426178928
'ambition': 0.0006907269901571533
```

In other words, your perception of your own attractiveness (on a scale from 1-10) had LESS than a 1% effect on if another person will want to date you… in THIS model. I would be happy to go out and say that you are likely more attractive than you are but I wanted to verify these numbers so I tried the same dataset with lab3's bayesian classifier and got these numbers

```
Accuracy = 0.5600069072699015
'attractive': 0.008737696425487862
'sincere': 0.002728371611120728
'intelligence': 0.0026938352616128645
'funny': 0.007097219823864642
'ambition': 0.002521153514073582
```

Both models had similar accuracies and feature importances so I have come to the conclusion that maybe we are all better than we think at dating.

I also performed and compared the two models using the titanic dataset:

```
Gaussian Bayesian Classifier:
0.6373626373626373
{'Pclass': 0.043956043956043946, 'Age': 0.02564102564102566, 'Fare': 0.014652014652014636}
```

```
Bayesian Classifier:
0.6821829855537721
{'Pclass': 0.06613162118780101, 'Fare': 0.025200642054574684, 'Age': 0.023916532905296994}
```

ETHICS:
The results of the analysis of the speed dating dataset were found to be a pretty positive answer for the world to hear. However, if the results had been skewed the other way, meaning self image has a large effect on whether or not you will get a date, the results can be negative to those who struggle with anxiety, mental health, and possibly those with disabilities as well. Let's say self image and probability were positively related. Then in that case, those who have less control over their self image will pretty much be told that they will not be getting a date easily. That is not something many people want to hear and is usually not a great predictor of datability anyways. I think the results of the analysis done here would be ok to release, giving inspiration to those who think they might not have what it takes, but if the results were the opposite, I likely would not release the results. Doing so would likely offend a sizeable portion of the population and might be seen as insensitive.