CS373 HW4

Due Date: April 20, 11:59 PM

Instructions for submission: In this programming assignment you will implement the Perceptron and the Naive Bayes algorithms and evaluate them on the Yelp dataset. Instructions below detail how to turn in your code and assignment on `data.cs.purdue.edu`.

You need to implement this assignment from scratch in Python. Do not use any already implemented models like those in the scikit-learn library. Also, programs that print more than what is required will be penalized. Your code needs to run in the server: `data.cs.purdue.edu`. Check that it works there by executing your script from the terminal.

For the coding implementations, submit your code as a Python program. For all other parts write your answers in a single PDF. Name this `yourusername_hw4.pdf`.

As usual, label the plots with the question number. Your homework **must** contain your name and Purdue ID at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized.

To submit your assignment, log into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely) and follow these steps:

1. Make a directory named `yourusername-hw4` (all letters in lower case) and copy your PDF file and Python file inside it. To do it remotely, use:

   ```
   scp ./path/to/your-file.pdf your-id@data.cs.purdue.edu:./remote/path/from-home-dir/
   ```

2. Go to the directory containing `yourusername-hw4` (e.g., if the files are in /homes/dan/dan-hw4, go to /homes/dan), and execute the following command:

   ```
   turnin -c s373 -p hw4 yourusername-hw4
   ```

   (e.g. Dan would use: `turnin -c s373 -p hw4 dan-hw4` to submit his work)

   Note that `s373` is the course name for turnin. **It is not a typo.**
3. To overwrite an old submission, simply execute this command again.
4. To verify the contents of your submission, execute the following command:

   ```
   turnin -v -c s373 -p hw4
   ```

0 Specification

0.1 Dataset Details

You will use `yelp_cat.csv` for this assignment. This data set is part of the Yelp database (similar to the one you worked with in hw2). The continuous variables have been removed and only categorical variables are retained. The dataset has 20,000 rows and 15 discrete attributes.

Fetaures: Consider the first 14 discrete attributes in `yelp_cat.csv` for $\mathbf{X}$.
Class label: Consider the attribute `goodForGroups` as the class label, i.e., $Y \in \{0, 1\}$.

0.2 Code Details

You are only allowed to use Python 2.7. You need to submit **three** python scripts `vanilla.py, avg.py` and `nbc.py`. Each of your python scripts should take the following arguments:

1. *trainingDataFile*: corresponds to a subset of the Yelp data (in the same format as `yelp_cat.csv`) that should be used as the training set in your algorithm.
2. *testingDataFile*: corresponds to another subset of the Yelp data (in the same format as `yelp_cat.csv`) that should be used as the test set in your algorithm.

Note: Your perceptron models (`vanilla.py, avg.py`) also need to take an additional parameter corrseponding to the maximum number of iterations it should run. More details mentioned in the perceptron section.

Your code should read in the training/test sets from the csv files, learn the respective model from the training set, apply the learned model to the test set, and evaluate the predictions with zero-one loss. Details on zero-one loss are given in the evaluation section.

Your input and output should look like this:

```
$ python vanilla.py train-set.csv test-set.csv 5
ZERO-ONE LOSS=0.2305

$ python avg.py train-set.csv test-set.csv
ZERO-ONE LOSS=0.2305

$ python nbc.py train-set.csv test-set.csv
ZERO-ONE LOSS=0.2305
```

Please stick to the given format. If the output is not in the correct format you **will** be penalized.

1 Perceptron (40 Points)

1.1 Perceptron Details (20 Points)

1. The perceptron model as we saw in class is given as:

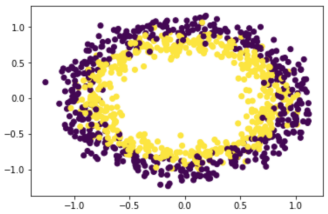
$$f(x) = \begin{cases} 1, & \sum w_j x_j \geq 0 \\ 0, & \sum w_j x_j < 0 \end{cases}$$

There is an important term called the *bias* missing from the equation above. Write the equation for the model with the bias term included. What is the significance of this term? Explain this using example plots for both cases.

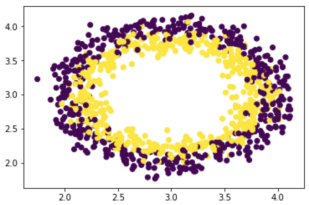
$$f(x) = \begin{cases} 1, & \sum w_j x_j + b \geq 0 \\ 0, & \sum w_j x_j + b < 0 \end{cases}$$

BIas is required for translation of the hyperplane, else all the hyperplanes would cross through the origin.

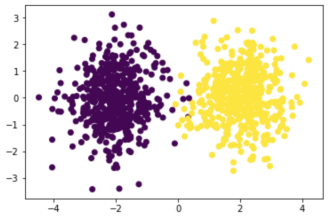
2. Below there are four figures that show the distribution of datapoints with binary classes. The two colors denote the different classes. For each of these, reason whether the following do would give a high(> 0.95) classification accuracy.
(i) a perceptron without bias
(ii) a perceptron with bias



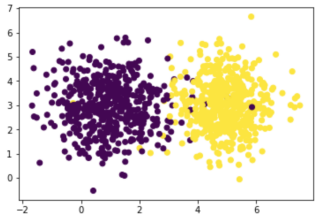
(a) None would give a good accuracy because data not linearly seprable(assuming no transformation)



(b) None would give a good accuracy because data not linearly seprable(assuming no transformation)



(c) Both (i) and (ii), since the data is linearly seprable



(d) Only (ii), since in (i) hyperplane needs to pass through origin

3. For the given Yelp data suggest a way of engineering a new feature from the given set(or subset) of features. Explain your answer and justify how this new feature might be capturing some form of complex behavior.
   This is an open ended question, so as long as your idea and justification make sense you should get full points.

4. The pseudocode for the vanilla perceptron algorithm is given below (Algorithm 1). What do the weights and bias terms denote? Why do we need to update these in a perceptron? How do line 8 and line 9 in the pseudocode achieve this? (**Hint:** Assume binary classes (0,1))
   The weights and bias denote the parameters of the perceptron model. The weights are used for roation of the hyperplane, while bias is used for the translation. The goal of the update is to adjust the parameters so that they are âĂIJbetterâĂİ for the current example. The perceptron is error-driven, thus it only makes updates when there is an error. In case of binary labels, the error is either -1 or 1. Error is -1(or 1) when the true label was 0(or 1), but $f(x)$ returned 1(or 0). Thus in this case we need to decrease(or increase) the weight and bias terms and move the activation in the proper direction which is acheived by lines 8 and 9 of the pseudocode.

5. **Average Perceptron** In an average perceptron, instead of returning the updated

Algorithm 1 Vanilla Perceptron

function TRAIN($\mathbf{D}, MaxIter$)
  $w_i \leftarrow 0$, for all $i = 1, \ldots, n$ ▷ Initialize Weights
  $b \leftarrow 0$ ▷ Initialize Bias
  $avg\_w_i \leftarrow 0$, for all $i = 1, \ldots, n$ ▷ Initialize Average Weights
  $avg\_b \leftarrow 0$ ▷ Initialize Average Bias
  max = MaxIter*len(D)
  step = MaxIter*len(D)
  for $iter = 1, \ldots, MaxIter$ **do**
    for all $(x, y) \in \mathbf{D}$ **do**
      $error \leftarrow y - f(x)$
      if $error$ **then**
        $b \leftarrow b + error$ ▷ Update Bias
        $w_i \leftarrow w_i + (error \times x_i)$, for all $i = 1, \ldots, n$ ▷ Update Weights
        $avg\_b \leftarrow avg\_b + error \times (step/max)$
        $avg\_w_i \leftarrow avg\_w_i + (step/max)(error \times x_i)$, for all $i = 1, \ldots, n$
      step = step - 1
  return $avg\_b, avg\_w_0, \ldots, avg\_w_n$
function PREDICT($b, w_0, \ldots, w_n, \hat{x}$)
  return f(x)

weights/bias at the end of the training, we return the average of all the weights/bias calculated in each iteration for each data point.

a) Write a pseudocode for the **Average Perceptron**. (**Hint:** You only need to add a few lines to the given pseudocode of the vanilla perceptron.)

b) What is the advantage/disadvantage of doing this?
Advantage: Better generalization. The weights of a vanilla perceptron are affected more by later points.

1.2 Perceptron Implementation (20 Points)

1. Implement the vanilla perceptron classifier. Name this file **vanilla.py**.
2. Implement the average perceptron classifier. Name this file **avg.py**.

Data: To make your Perceptron work, you will need to create binary features(similar to what you did in the last homework). You can assume you will not see nay new attribute value other than what is in the data provided to you.
Hint: To create binary features, you could use
`pd.get_dummies(X, columns=X.columns.values)`.
Please note here `X` needs to be a data frame. Once you have converted it to binary

features, you can convert it to a matrix as you did in the last homework.
Note: As mentioned earlier, the two scripts need to take an additional argument $MaxIter$ corrseponding to the maximum number of iterations it should run. Set the default value of $MaxIter = 2$ in the case no **third** argument is specified.

2 Naive Bayes (40 Points)

Smoothing: For this assignment, you will use Laplace smoothing in the parameter estimation. For an attribute X_i with k values, Laplace correction adds 1 to the numerator and k to the denominator of the maximum likelihood estimate for $P(X_i = x_i|Y)$.

2.1 Naive Bayes Details (20 Points)

1. Write down the mathematical expression for $P(Y|X)$ given by the Naive Bayes Classifier(**Hint:** It's not just the Bayes equation).

$$\begin{aligned} P(Y|X) &= \frac{P(X|Y)P(Y)}{P(X)} \\ &= P(Y)\frac{\prod_i P(X_i|Y)}{\sum_i \prod_j P(x_j|y_i)} \\ &\propto P(Y)\prod_i P(X_i|Y) \end{aligned}$$

2. Suppose your data has binary class labels , i.e $y \in \{0, 1\}$, write the expression for predicting the class for a given input row. You will use this expression in your implementation.

$$\hat{y} = \begin{cases} 0 & \text{if } P(X|y=0) \cdot P(y=0) > P(X|y=1) \cdot P(y=1) \\ 1 & \text{otherwise} \end{cases}$$

3. State the naive assumption that lets us simplify the expression $P(X|Y)P(Y)$. Is this assumption true for the given Yelp data? Explain why or why not?

   The niave assumption is that each feature x_i is conditionally indepentedent of every other feature x_j. This allows us to express $P(X|Y)$ as $\prod_i P(X_i|Y)$ by using the product rule for independent events. This assumption does not hold for the current data set since we have features that are not conditionally independent. e.g. Alcohol and attire are not conditionally independent given goodForGroups.

   $P(attire = casual|goodForGroups = 1) = 0.871440$
   $P(alcohol = full\_bar|goodForGroups = 1) = 0.376103$
   $P(attire = casual|goodForGroups = 1) \cdot P(alcohol = full\_bar|goodForGroups =$

1) = 0.32775
$P(attire = casual, alcohol = full\_bar|goodForGroups = 1) = 0.284513$

Clearly, the two are not independent.
$P(attire = casual, alcohol = full\_bar|goodForGroups = 1) \neq$
$P(attire = casual|goodForGroups = 1) \cdot P(alcohol = full\_bar|goodForGroups = 1)$

4. What part of the expression corresponds to the class prior? Considering the entire Yelp data as the training dataset, calculate the maximum likelihood estimate for the class prior with and without smoothing. What is the effect of smoothing on the final probabilities?

   The class prior is P(Y) Without smoothing: $P(goodForGroups = 0) = 0.3995$
   $P(goodForGroups = 1) = 0.6005$ With smoothing $P(goodForGroups = 0) = 0.39951$
   $P(goodForGroups = 1) = 0.60049$

   On the Yelp dataset, the change is very small. Theoretically, smoothing assigns a non-zero probability to the class label being anything other than those seen in the training dataset.

5. Specify the full set of parameters that need to be estimated for the NBC model of the Yelp data. How many parameters are there?

   We need to estimate

   $$P(X_i = x_i|Y = y_i)$$

   where X_i is a feature, x_i represents a unique value of X_i and y_i represents the class label. For the Yelp dataset, as shown below there are 307 unique values. We need to computer $P(X_i = x_i|Y = y_i)$ for each unique value x_i of each feature X_i conditioned on a class label y_i. This sums upto $307 \times 2 = 614$ CPD values. We also need to estimate priors $P(Y = 1)$and $P(Y = 0)$. Therefore, total number of parameters = 616. Note that using the properites of probability we dont need to calculate all these values, we need to calculate the CPD for (number of distinct values - 1). The final probability can be calculated by subtracting the sum from 1. Thus $293 \times 2 + 1 = 587$ is also a valid answer.

Attribute	#Distinct Values
city	246
state	14
stars	9
open	2
alcohol	3
noiseLevel	5
attire	4
priceRange	5
delivery	3
waiterService	3
smoking	4
outdoorSeating	3
caters	3
goodForKids	3

6. For the Yelp data, explicitly state the mathematical expression for the maximum likelihood estimates (with smoothing) of the CPD parameters for the attribute `alcohol` conditioned on the the class label `goodForGroups`.
   Alcohol has 3 possible values: None, full_bar, beer_and_wine
   goodForGroups has 2 distinct values(0 and 1).
   $P(alcohol = a|goodForGroups = g) = \frac{|goodForGroups=g\&priceRange=p|+1}{|goodForGroups=g|+3}$

7. Consider the entire Yelp data as the training dataset and `goodForGroups` as the class label. Estimate the conditional probability distributions of the following attributes with and without smoothing:
   a) stars
   Without

```
stars  goodForGroups
1.0    0                0.004881
1.0    1                0.001832
1.5    0                0.021277
1.5    1                0.011324
2.0    0                0.046683
2.0    1                0.036053
2.5    0                0.094869
2.5    1                0.096503
3.0    0                0.159574
3.0    1                0.193672
3.5    0                0.223780
3.5    1                0.283930
4.0    0                0.240676
```

```
4.0    1                0.257286
4.5    0                0.165582
4.5    1                0.104163
5.0    0                0.042678
5.0    1                0.015237
```

With:

```
stars  goodForGroups
1.0    0                0.005004
1.0    1                0.001914
1.5    0                0.021392
1.5    1                0.011402
2.0    0                0.046787
2.0    1                0.036120
2.5    0                0.094951
2.5    1                0.096543
3.0    0                0.159628
3.0    1                0.193668
3.5    0                0.223804
3.5    1                0.283886
4.0    0                0.240693
4.0    1                0.257253
4.5    0                0.165633
4.5    1                0.104200
5.0    0                0.042784
5.0    1                0.015314
```

b) waiterService

Without:

```
waiterService  goodForGroups
BLANK          0                0.621527
BLANK          1                0.181349
FALSE          0                0.149186
FALSE          1                0.337635
TRUE           0                0.229287
TRUE           1                0.481016
```

With:

```
waiterService  goodForGroups
BLANK          0                0.621559
BLANK          1                0.181405
```

FALSE	0	0.149289
FALSE	1	0.337668
TRUE	0	0.229377
TRUE	1	0.481027

c) caters

Without:

caters	goodForGroups	
BLANK	0	0.754944
BLANK	1	0.426561
FALSE	0	0.134668
FALSE	1	0.298585
TRUE	0	0.110388
TRUE	1	0.274854

With:

caters	goodForGroups	
BLANK	0	0.754956
BLANK	1	0.426580
FALSE	0	0.134773
FALSE	1	0.298623
TRUE	0	0.110497
TRUE	1	0.274896

d) attire

Without:

attire	goodForGroups	
BLANK	0	0.597121
BLANK	1	0.094671
casual	0	0.391489
casual	1	0.871440
dressy	0	0.010638
dressy	1	0.031224
formal	0	0.000751
formal	1	0.002664

With:

attire	goodForGroups	
BLANK	0	0.597127
BLANK	1	0.094735
casual	0	0.391536
casual	1	0.871349

```
dressy  0          0.010761
dressy  1          0.031301
formal  0          0.000876
formal  1          0.002747
```

What is the effect of smoothing (e.g., any difference compared to 4.)? Which attribute shows the most association with the class? Why?

Smoothing doesn't have much effect, similar to part 4, because here we do not have any attribute value that has a probability 0. Attire shows the most association, since given goodForGroups=1, the probability that attire=causal is highest.

2.2 Naive Bayes Implementation(20 Points)

1. Implement a Naive Bayes classifier in python. Name this file **nbc.py**.
   Data: Since the Naive Bayes can handle multi-valued discrete attributes, there is no need to create binary features. Here you cannot assume that you may not see a new attribute value other than what is being provided to you.
   Hint: To avoid underflow issues you could use **log** values.
   Note: Your classifier should be able to handle attribute values that appear in test data but were not observed in the training data. One way to do this is have a separate label to accommodate not-in-training labels for all attributes that can have a variable number of values.

3 Analysis(20 Points)

Now you will evaluate your models **avg.py** and **nbc.py** using cross validation and learning curves. Cross-validation is a powerful tool to tune your hyperparameters and gauge how well your learned model generalizes. The basic idea behind cross-validation is that a subset of the data is kept aside(called the validation set) before the training begins. When we have a learned model, the validation set can used to test the performance of our model.

In k-fold cross-validation, the data is divided into k subsets. In each iteration we use one such subset as the test set and train on the remaining $k-1$ subsets as training set. This process is repeated k times. We then take the average error accross all k trails. This gives a more accurate measure of model quality than if you were to hold out some fixed subset of the training data as validation set.

A learning curve is a way to plot the evolution of the loss as size of the training set

changes. To evaluate the learned models you will use the following loss functions:

1. Zero-One Loss (for both models)
2. Squared Loss (for Naive Bayes)

Let $y(i)$ be the true class label for example i and let $\hat{y}(i)$ be the prediction for i. Then zero one loss for the test dataset T of n instances is:

$$Loss_{0/1} = \frac{1}{n} \sum_{i \in n} \left\{ \begin{array}{ll} 0 & \text{if } y(i) = \hat{y}_i \\ 1 & \text{otherwise} \end{array} \right\}$$

Let p_i refer to the probability that the Naive Bayes classifier assigns to the true class of example i (i.e.,$p_i := p(\hat{y}(i) = y(i))$. If $p_i > 0.5$, the prediction for example i will be correct (i.e., $\hat{y}(i) = y(i)$), but otherwise if $p_i < 0.5$, the prediction will be incorrect (i.e.,$\hat{y}(i) \neq y(i)$). Then the squred loss for the test dataset T of n instances is:

$$Loss_s q(T) = \frac{1}{n} \sum_{i \in n} (1 - p_i)^2$$

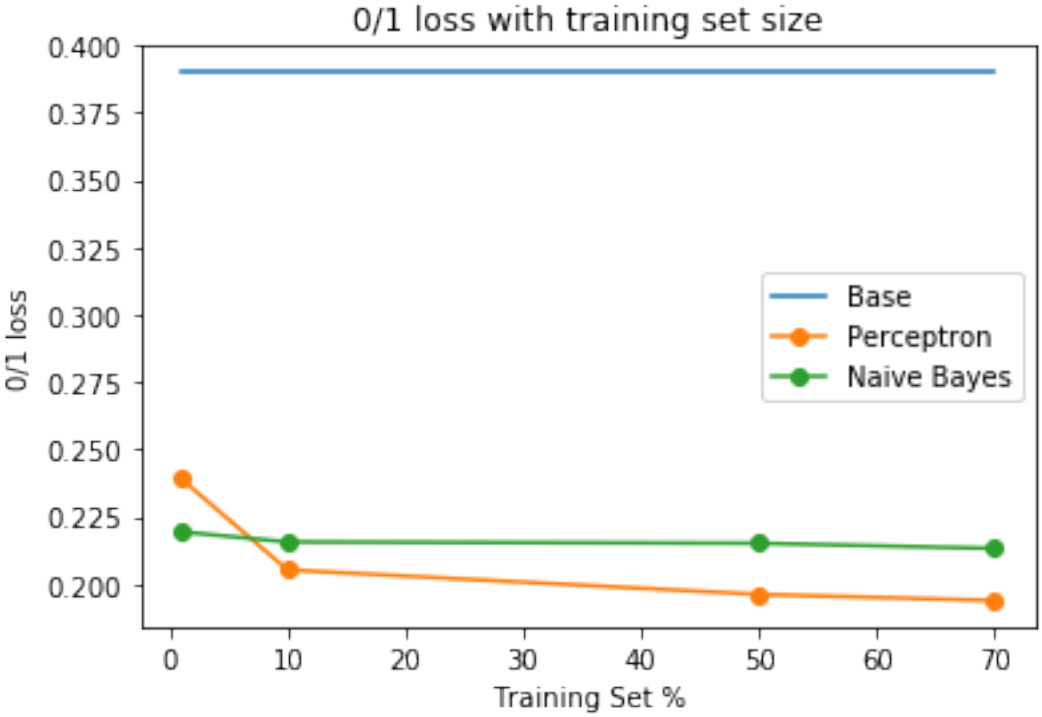
In general you would use cross-validation to tune the hyperparameters of your model (e.g. $MaxIter$ in the Perceptron model), but for the following problems you should use the default value.

1. For each $\%$ in $[1, 10, 50, 70]$:
   For i in [0...9]:
   - Randomly sample $\%$ of the data to use as the training dataset.
   - Use the remaining $(100 - \%)$ of the data as the test dataset.
   - Learn a model from the training data and apply it to the test data.
   - Measure the loss on the test data.

   a) Record the **mean** zero-one loss observed across the ten trials for each training set size (i.e., sample $\%$) for both Naive Bayes and Average Perceptron classifiers.
   Perceptron :0.23905,0.20534,0.19624,0.19398
   Naive Bayes:0.21944,0.2157,0.21522,0.21328

   b) Record the **mean** squared loss across the ten trials for each training set size for the Naive Bayes classifier.
   0.18748,0.18706,0.18302,0.18134

2. a) Plot a learning curve of training set size vs. zero-one-loss for both Naive Bayes and Perceptron.
   b) To the above plot add the baseline default error that would be achieved if you just predicted the most frequent class label in the overall data.



c) Discuss your results:

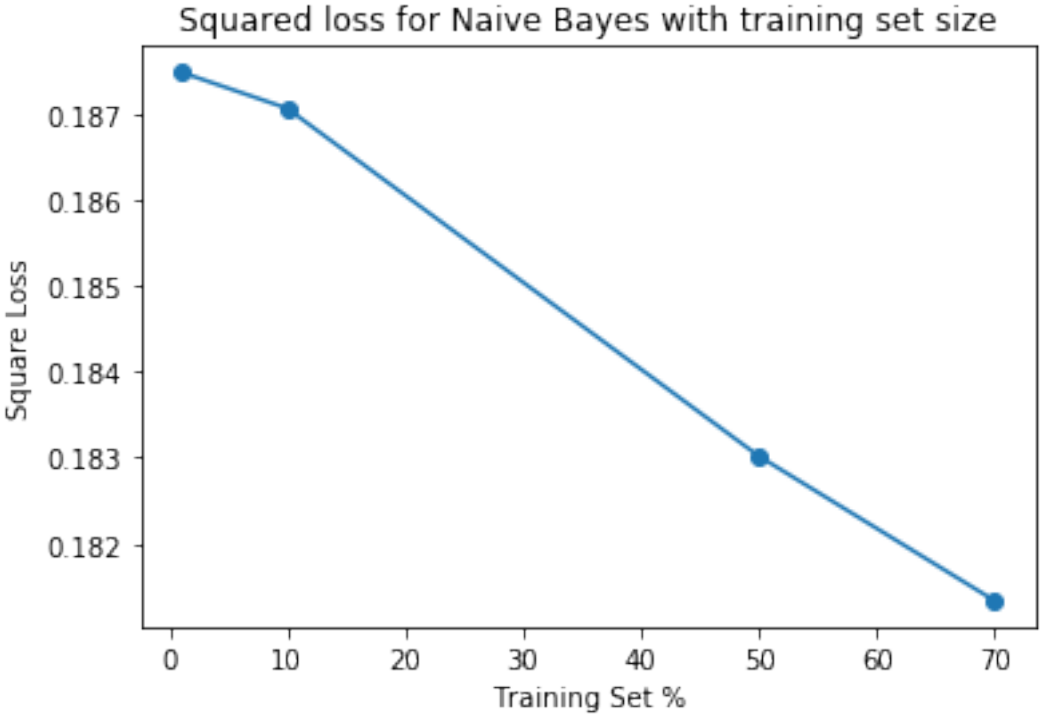
How the two classifiers compare with each other and with the baseline.

Both perform significantly better than baseline, and are comparable to each other.

How is zero-one loss impacted by training set size.

Generally it tends to go down, but here we see it stays almmost constant for Naive Bayes, and for perceptron stabalizes after a drop from 1 to 10.

3. a) Plot a learning curve of training set size vs. square-loss for Naive Bayes.



b) Discuss how zero-one loss performance compares to square-loss. Which do you think is a better metric and why?
Square-loss is consistently lower than zero-one-loss. This is because of the way both the losses are computed: zero-one-loss penalizes a miss by 1 whereas square-loss uses the prediction probability and adds the square of the distance by which the NBC prediction is off (i.e.,$(1 - p(trueClass))^2$). We could say square loss is a better metric since it gives a magnitude to the distance from the true prediction.