Project 1 Report
Holten Bradley (bradl515)

Experiment 1

This data set includes 3333 clients with 21 features each. The attributes have a variety of types ranging from numeric, Boolean, and to an object. The features are shown in Figure 1. The features labeled object are usually binary. State and Phone number are discrete objects because they are labels while International Plan and Voice Mail Plan are binary objects because they are yes or no so can/should be considered binary. The Boolean feature listed in Figure 1 is considered our class for the clients being either loyal or not loyal. This is also a binary feature.

Figure 1

#	Column	Non-Null Count	Dtype
0	state	3333 non-null	object
1	account length	3333 non-null	int64
2	area code	3333 non-null	int64
3	phone number	3333 non-null	object
4	international plan	3333 non-null	object
5	voice mail plan	3333 non-null	object
6	number vmail messages	3333 non-null	int64
7	total day minutes	3333 non-null	float64
8	total day calls	3333 non-null	int64
9	total day charge	3333 non-null	float64
10	total eve minutes	3333 non-null	float64
11	total eve calls	3333 non-null	int64
12	total eve charge	3333 non-null	float64
13	total night minutes	3333 non-null	float64
14	total night calls	3333 non-null	int64
15	total night charge	3333 non-null	float64
16	total intl minutes	3333 non-null	float64
17	total intl calls	3333 non-null	int64
18	total intl charge	3333 non-null	float64
19	customer service calls	3333 non-null	int64
20	churn	3333 non-null	bool

Now the numerical features are not going to be binary. Account Length is Discrete since it is not considered to have a decimal value. Area code is also discrete along with number of vmail messages, total eve calls, total eve charge, total night calls, total night charge, total intl calls, total intl charge, and total customer services calls. Total day minutes is continuous since it is possible to have a fraction of a minute. This is concurrent with total minutes for eve, night, and intl. These values are able to have a fraction of a minute while cost and calls have a limited number since at most two decimals are allowed.

There are no missing values in this dataset. This does not mean that all features are necessary. The irrelevant features have close averages in both the classes. This will tell us that there is no clear discrepency and it is not giving us enough info to make a clear decision unless we want a series of rules. These numerical features include area code and total day, eve, night, and intl calls.

The mean, standard deviation, median, maximum, and minimum for the minutes are found in Figure 2.

Figure 2

	total day minutes	total day charge	total eve minutes	total eve charge	total night minutes	total night charge	total intl minutes	total intl charge
count	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00
mean	179.78	30.56	200.98	17.08	200.87	9.04	10.24	2.76
std	54.47	9.26	50.71	4.31	50.57	2.28	2.79	0.75
min	0.00	0.00	0.00	0.00	23.20	1.04	0.00	0.00
25%	143.70	24.43	166.60	14.16	167.00	7.52	8.50	2.30
50%	179.40	30.50	201.40	17.12	201.20	9.05	10.30	2.78
75%	216.40	36.79	235.30	20.00	235.30	10.59	12.10	3.27
max	350.80	59.64	363.70	30.91	395.00	17.77	20.00	5.40

The total charges can be ignored since I will be considering them as discrete with the max amount of 2 decimal places. There is no such thing as 0.1 of a cent. This means 0.01 is the smallest possible value for cost making it discrete.

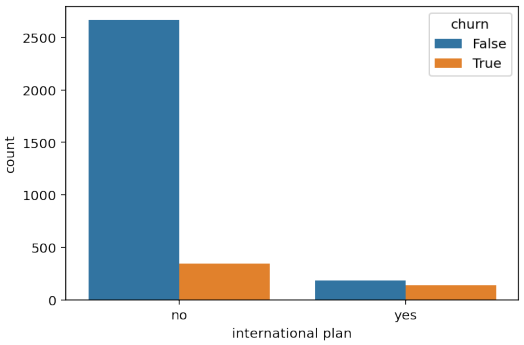
Our data is coming from 51 different states. All 50 states and the District of Columbia. Within these 51 states, 86% of the clients are loyal. They have not churned. Whereas we are left with 14% of the clients who are churned and have left the company. These results are skewed to the loyal clients. Of the loyal clients, the average number of customer service calls is 1.45. Of the non-loyal clients, the average number of customer service calls is 2.23. The average for the whole data set is 1.56. This means the users who have utilized the customer service line have found it to be helpful with the churn results skewed towards the loyal users.

The total day charge is another feature that deserves some attention. The Max day charge is 59.64$ while the Min day charge is 0$ since there are clients who don't use any day minutes. The average day charge is 30.56$. If we order the data in ascending order, we find that day charge is much more relevant to churn than imagined. In ascending order, the lowest day charges are shown and 9/10 of the bottom clients are loyal while the top ten are all already gone and have churned.

Churned		**Non-Churned(Loyal)**	
account length	102.66	account length	100.79
area code	437.82	area code	437.07
number vmail messages	5.12	number vmail messages	8.60
total day minutes	206.91	total day minutes	175.18
total day calls	101.34	total day calls	100.28
total day charge	35.18	total day charge	29.78
total eve minutes	212.41	total eve minutes	199.04
total eve calls	100.56	total eve calls	100.04
total eve charge	18.05	total eve charge	16.92
total night minutes	205.23	total night minutes	200.13
total night calls	100.40	total night calls	100.06
total night charge	9.24	total night charge	9.01
total intl minutes	10.70	total intl minutes	10.16
total intl calls	4.16	total intl calls	4.53
total intl charge	2.89	total intl charge	2.74
customer service calls	2.23	customer service calls	1.45
churn	1.00	churn	0.00

We can compare the averages of the Churned and Non-Churned clients to make some assumptions about the data. Our loyal customers seem to use their phone less than those who have churned. Loyal customers have less day minutes, less eve minutes, and less night minutes. I believe we would benefit from lowering our day minutes rate and making up the difference in the night fare. The day fare for the churn clients are much higher than the loyal ones. I do not believe our churned are concerned with where they are so we should not be concerned with area code. We may need to improve our customer service if there has to be multiple calls to decrease the amount of customer service calls. The average data shows us that minutes is more important than calls and the largest discrepancy may be in the day minutes/cost. As a company, this is where I would start.

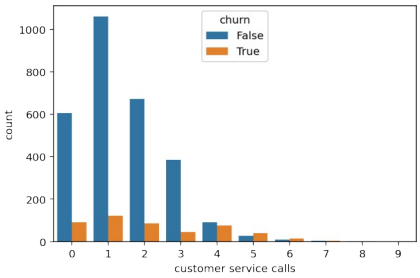
If we were to have a model that stated: international plan = "no" then churn = "False", we would get TP = 2664, FN = 346, FP = 186, and TN = 137. This would give us a precision of 0.885, a recall of 0.935, and an accuracy of 0.840. This is a generally accurate model and should be considered in the company.



P (churn = True | international plan = 'yes') = 0.42
P (churn = False | international plan = 'yes') = 0.58
P (churn= True | international plan = 'no') = 0.11
P (churn = False | international plan = 'no') = 0.89
P (international plan = 'yes'/'no' | churn = True) = 0.284 / 0.716
P (international plan = 'yes'/'no' | churn = False) = 0.065 / 0.935

Customer Service calls was another feature that was concerning when comparing the averages of the two classes. Below we can see how many customer service calls were made and the corresponding number of individuals in each class.

customer service calls	0	1	2	3	4	5	6	7	8	9	All
churn											
False	605	1059	672	385	90	26	8	4	1	0	2850
True	92	122	87	44	76	40	14	5	1	2	483
All	697	1181	759	429	166	66	22	9	2	2	3333



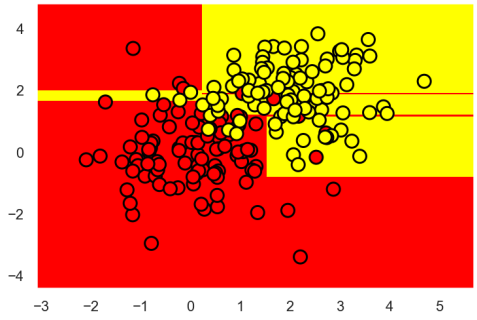
If the customer has made no call to customer service, the probability of them leaving is 0.132. The probability does not mainly rise with more customer service calls. With 1, 2, 3, 4, 5, 6, 7, 8, 9 calls respectively the probablilities of those leaving are 0.103, 0.115, 0.103, 0.458, 0.606, 0.636, 0.556, 0.5, and 1.0. This shows that once a customer makes that 4th call, they are way more likely to leave the company. Our customer service can not be too bad if they are making majority of the customers to stay after 1, 2 and 3 calls.
If we receive many service calls, more than 3, and we do have an international plan then that customer should leave. The accuracy on this model is 0.21, the recall is 0.498 and the precision is 0.43. This tells us that we might not believe this model and steer towards the opposite.

Experiment 2

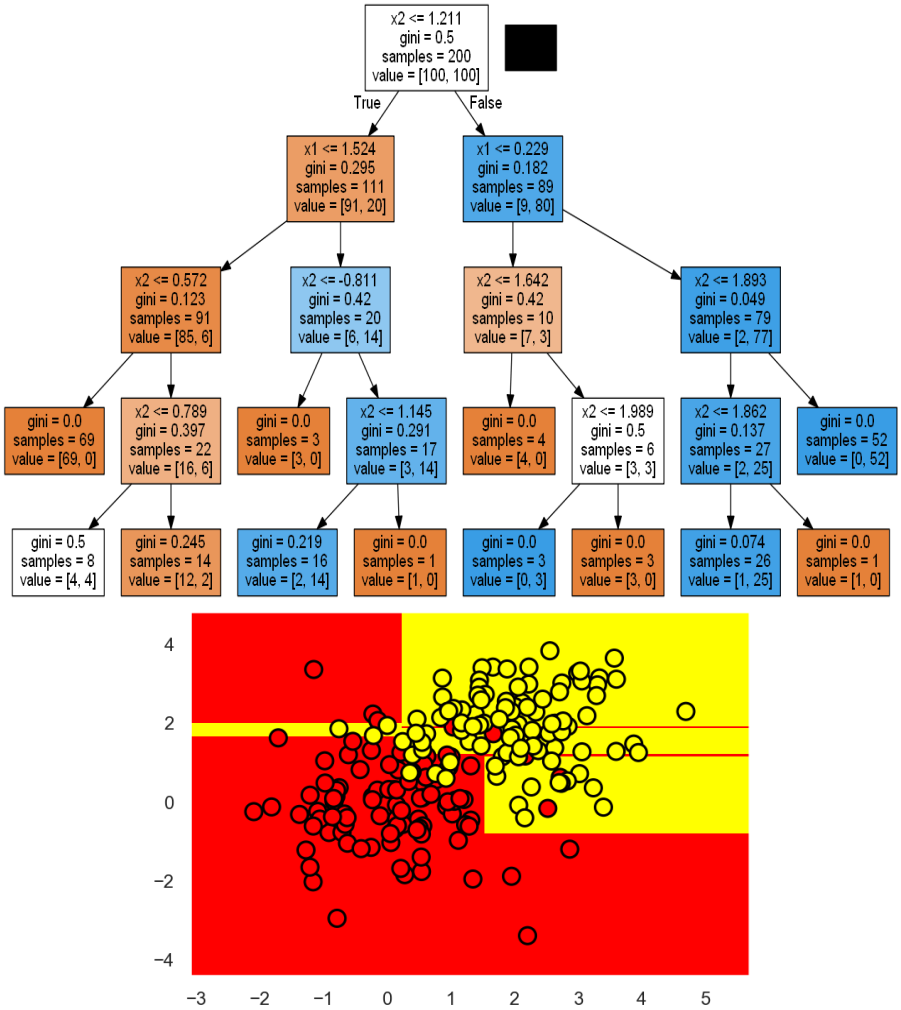
The decision tree classifier has max depth and criterion for the quality of the split. If we do not set these parameters we may do way more work than needed and make way too many splits. This could lead to overfitting and reviewing too much data for us to handle. It is also important to set the criterion in order to get an accurate split that is useful.

If we do not set the max depth, we will have a number of issues. For example, setting the max depth to 1 gives an example of an underfitting tree. This is almost as if we drew a line of best fit and called it an accurate representation. This would not be accurate. If we do nothing and have too high of a max depth. The tree becomes too complicated and it takes too many decisions to reach a leaf. This leads us to random jumps in our data. Figure 1 shows us that there are colors cutting into each other. This signifies overfitting.

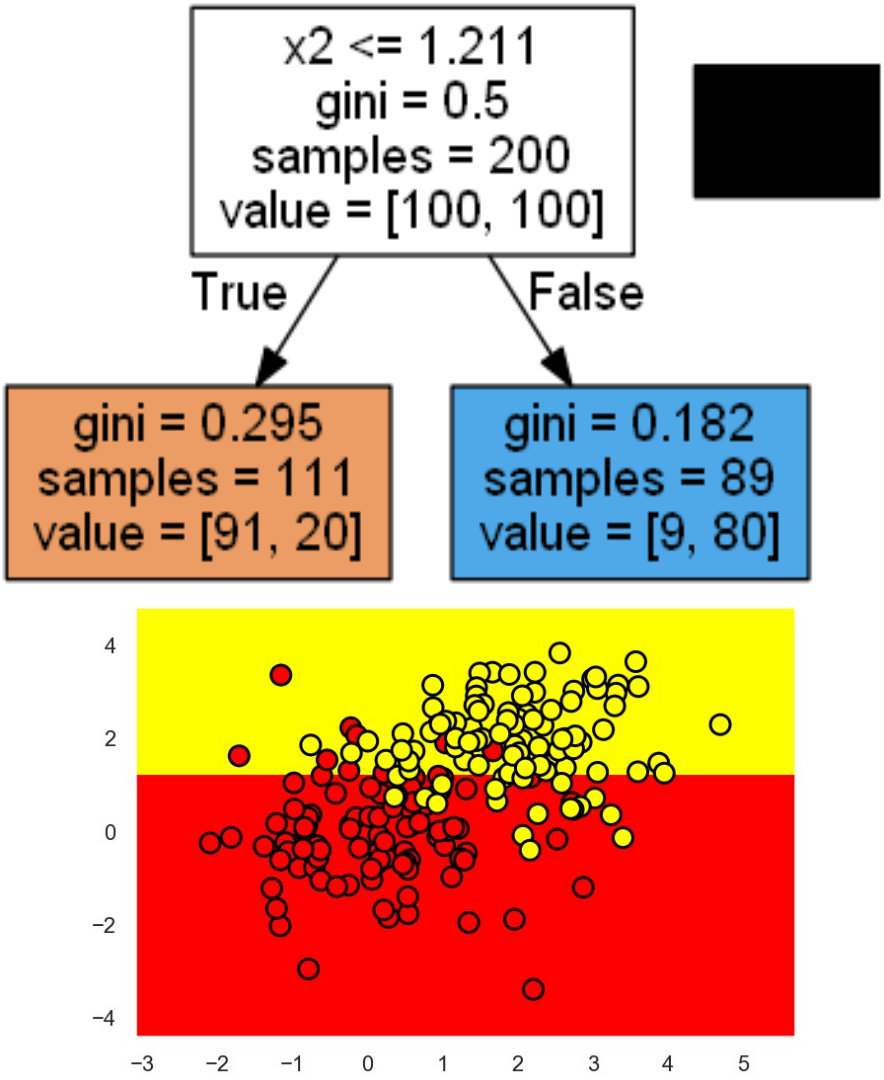
Figure 1



Max depth = 4



Max Depth = 1



For the Bank Dataset, the decision tree splits the ages into 5 different parts : 43.5, 19, 22.5, 30, and 32 years. These 5 different parts are exactly the mean values between the ages at which the target class "switches" from 1 to 0 or 0 to 1. To illustrate further, 43.5 is the average of 38 and 49 years; a 38-year-old customer failed to return the loan whereas the 49-year-old did.

**Given d = ([0,0], [0,0], [1,1], [1,1]) and put into the standard scalar, it gives us the transformation of array([[-1, -1], [-1, -1], [1, 1], [1, 1]]).

The experiments performed in the underfitting and overfitting section of this Experiment were a variation of different tree sizes against both the big and small data. We can see that the large trees were more accurate with the big data on the test set while the small decision tree has little difference with both test data sets. This means we want to have large data and large decisions trees to get the most accurate results. This is why big data has become so popular.

The next section of tests has to do with imbalanced and balanced classes. In this experiment we are testing training vs testing on imbalanced sets and balanced sets. If we train on a balanced set while testing on a unbalanced set we will get an accuracy of 0.7947 with a f1_score, precision, and recall of

0.05, 0.027, and 1.0. Next, we train on the balance set and test on a balance set with an accuracy of 0.7932 and f1_score, precision, and recall are 0.789, 0.805, and 0.774. Our accuracy is not much different while training on a balanced set. Now let's train on an imbalanced set. Training on an imbalanced set and testing on a balanced set gives us an accuracy of 0.5789 and a f1_score, precision, and recall of 0.2727, 1.0, and 0.15789. If we both train and test on an imbalanced set we get an accuracy of 0.9908 and a f1_score, precision, and recall of 0.0, 0.0, and 0.0. Which means training on an imbalanced and testing on an imbalanced set will give us the highest accuracy and very little amount of error compared to the rest of the testing.

Now we tested for irrelevant attributes. In testing for these attributes, we found accuracy of decision trees is hurt when adding irrelevant features. We have a training set accuracy of 1.0 in both test sets, but the test set with irrelevant features is impaired by almost 4%. This is why we must make sure we are not adding irrelevant features when making an assumption. It could damage our certainty.

For the customer churn prediction, we are told to add a max depth of 5 in the decision tree gives us an accuracy of 94%. What if we leave this to the default? We get an accuracy of 92%. The accuracy is falling because we have too many leaves which leads to some incorrect results. We are creating unnecessary branches which may lead some predictions to the incorrect result.

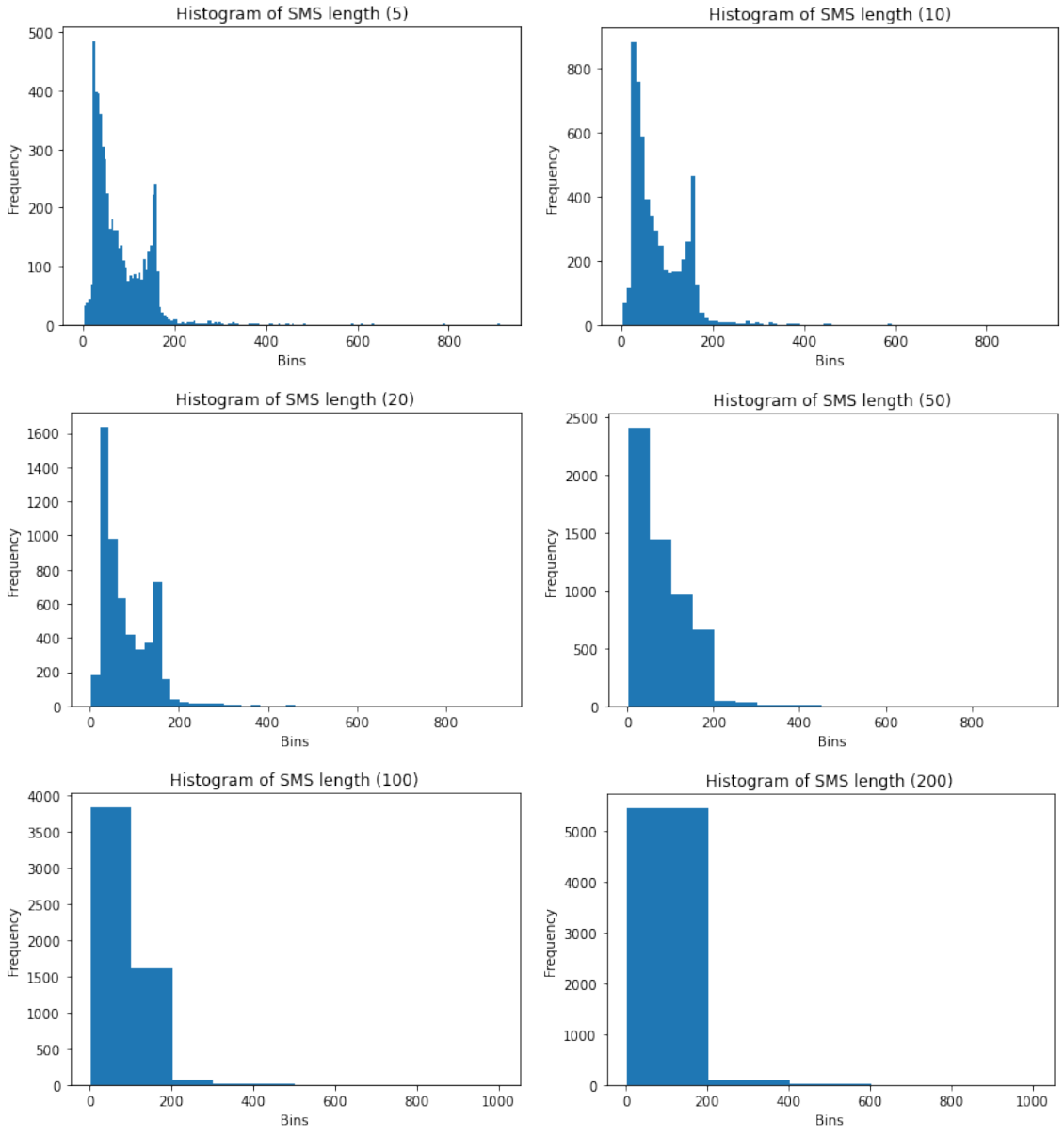
We construct 1500 fits with 10 fold and 150 candidates. With the same number of candidates and a 5-fold, we have 750 fits. Then adding the holdouts, we get 126 candidates with 1260 fits. We have to search double the amount of trees in the 10 fold cross-validation. The best choice for our parameters found is a max depth of 9 now and max features of 12.
For the K nearest neighbor algorithm with the 10-fold cross-validation finds the best choice for k is 9.

The final experiment to look at is the MNIST dataset. The max depth 5 tree has an accuracy of 0.667 and the K-Nearest neighbor with max neighbor 10 has an accuracy of 0.976. Now, we cross-validate the tree data to find the optimal parameters. With the 5-fold validation, we find the best parameters to be a max depth of 10 and max features of 50 to give us an accuracy of 0.843. This is better than the original but still not as good as the K-Nearest Neighbor.

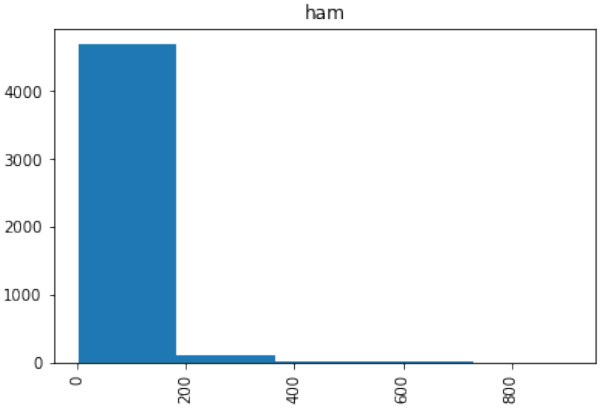
Experiment 3

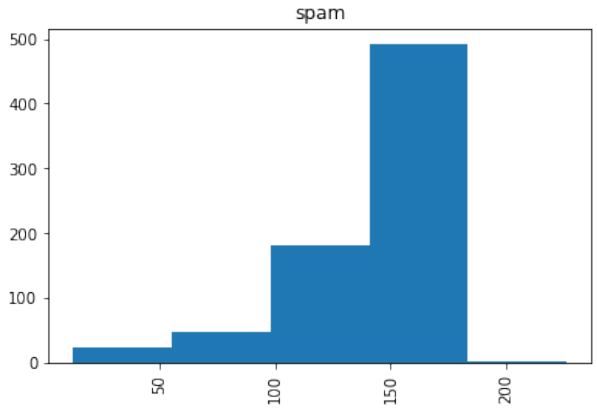
There are 5572 records with two features: {Spam, Ham}, {sms}. This is distributed unevenly with most messages being ham at 4825 and less are spam with 747. This is still a lot of spam messages even though we are skewed to the ham label. There are 5169 unique messages in the dataset. The most frequent sms was “Sorry, I’ll call later” which occurs 30 times.

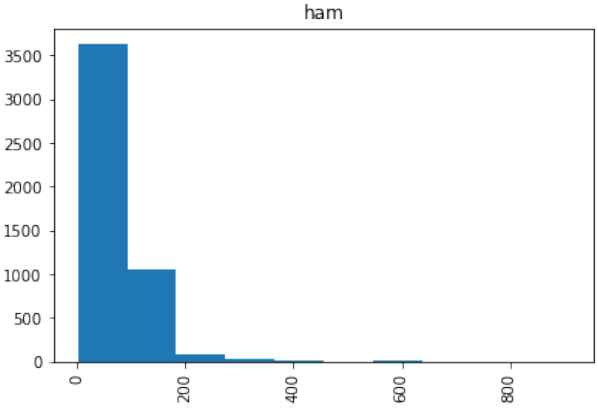
These messages are a variation of length from 910 to 2.

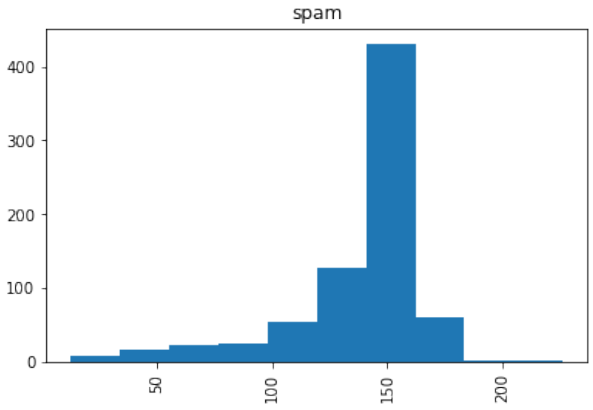


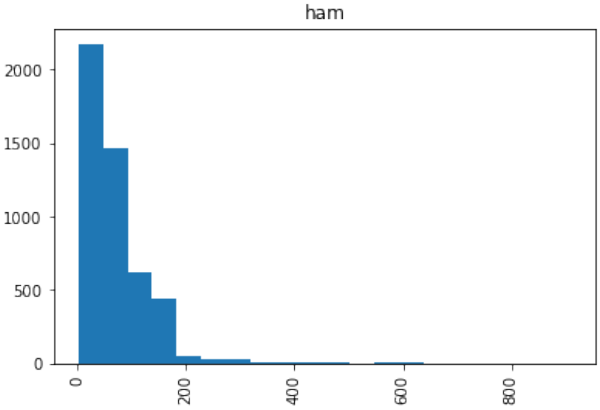
After viewing these plots, it is worth noting that majority of the text lengths are over 200. It is also known there are two spikes in the lower bin size plots. These spikes should be considered when trying to find spam sms. Below are the same plots but for each label.

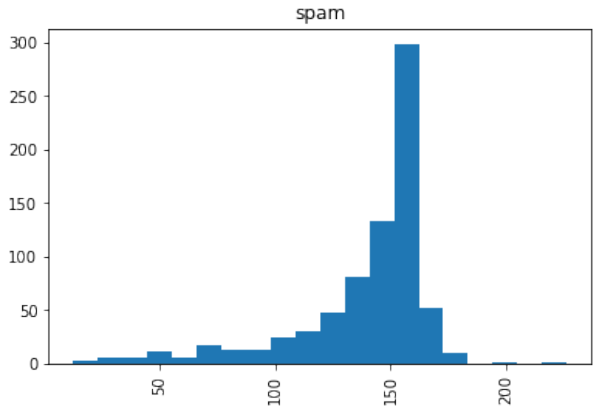

ham
4000
3000
2000
1000
0
0
200
400
600
800

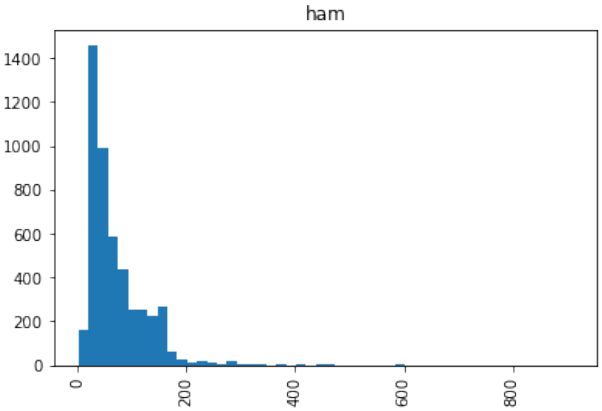

spam
500
400
300
200
100
0
50
100
150
200

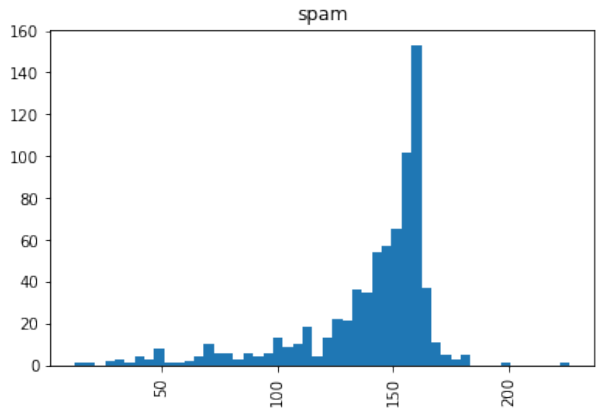

ham
3500
3000
2500
2000
1500
1000
500
0
0
200
400
600
800

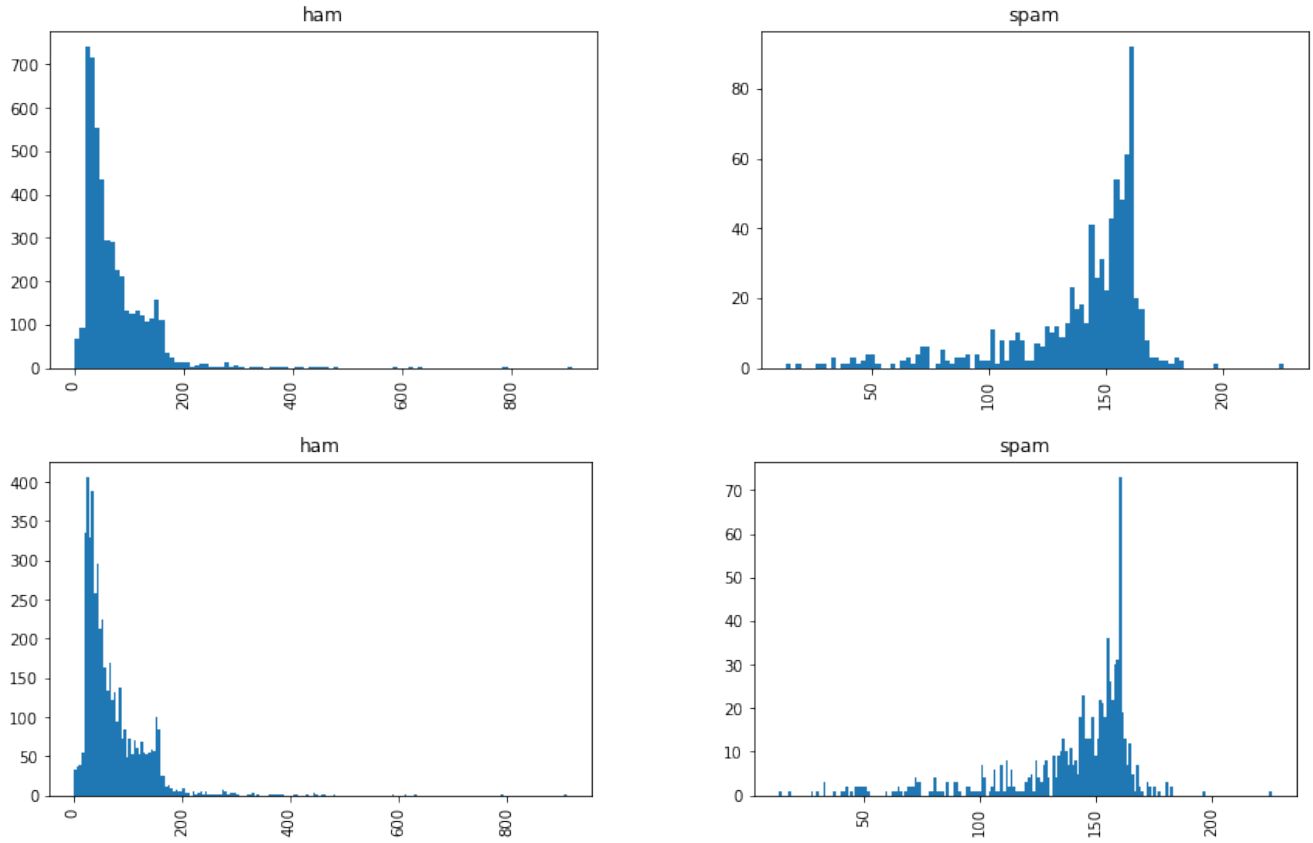

spam
400
300
200
100
0
50
100
150
200


ham
2000
1500
1000
500
0
0
200
400
600
800


spam
300
250
200
150
100
50
0
50
100
150
200


ham
1400
1200
1000
800
600
400
200
0
0
200
400
600
800


spam
160
140
120
100
80
60
40
20
0
50
100
150
200



These graphs show us the truth about the spam sms. The spam sms is most likely at around 160-170 characters long. The non-spam messages appear to be anywhere from 0 -200 with a spike at <100. I believe we have found the length to look for the best portion of spam sms.

Next, we have the bag of words approach which converts all words into tokens and counts the number of tokens in the ‘bag’. We have to get rid of all symbols, since they are not words. Then, we have to separate the words by the spaces. This will allow us to get each individual word and count the frequencies of each. We have to convert all these words to lowercase to make sure the comparison is not case sensitive. The same could happen if we made all characters uppercase as well.
This process is implemented in the CountVectorizer(). The CountVectorizer() also has stop words than will remove all words from the document that match a word in the stop words list. If we set this to English it will clear our sms messages of all English stop words which are defined in scikit-learn. Example of stop words are “sorry”, “is”, “are”, “a”, and “the”.

For a dataset, we can generate a document-term matrix by comparing the frequency with the word. For example, ‘are’ has a frequency of 1 in the first phrase/vector and 0 in the rest. Therefore, row 1 of ‘are’ is 1 while row 2 and 3 are 0. It is a frequency count of the word(col) in each vector(row). We have to separate the data into train and test before generating a document-term matrix. This is because we cannot test on data that has already been trained on. If we do, it will cause invalid results since the algorithm has already found the answer in training.

```
[[1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1],
 [0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 2, 0],
 [0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1]]
```

	are	call	from	hi	home	how	money	now	or	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	1	
1	0	1	1	0	1	0	1	0	0	2	0	
2	0	1	0	1	0	0	0	1	1	0	1	

This has 12 features but it is possible to reduce the amount of features by adding stop words to the CountVectorizer(). A pro is less features which would allow us to focus on more frequent features. A con of this strategy is we will have some missing values in our final results.

We use multinomial Naive Bayes implementation. This classifier is suitable for classification with discrete features. We have word counts for text classification so this would be discrete. Gaussian Naive Bayes would not be our optimal implementation for this data. This is because it is better suited for continuous data as it assumes that the input data has a normal distribution.

Diabetes Data

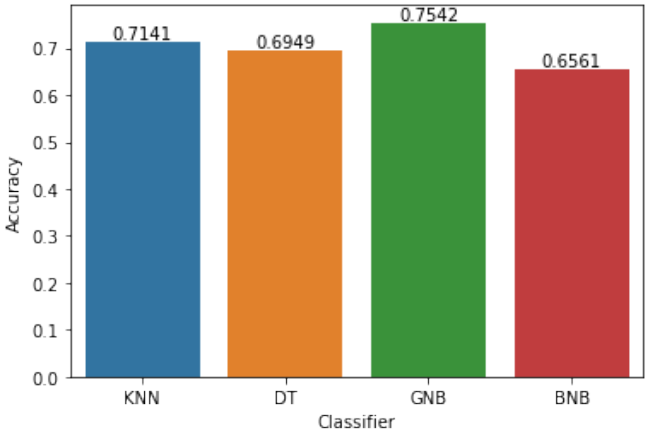
We have 768 Records with 9 features each. The features include : Pregnancies, glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age, and Outcome. Outcome is our two classes, with 1 being True and 0 being False. We have 500 of the patients without diabetes and 268 of the patients who do have diabetes. This is slightly skewed to the patients without diabetes. The features with the highest averages will be evaluated in our search for relevant feature. Glucose, Blood Pressure, and BMI have higher averages for the whole data set. We have no null data points, but this does not mean we do not have missing values. There are lots of unexpected outliers for Skin Thickness and Insulin. This outlier is a zero which most likely means data was not entered.

After looking at the individual outcome graphs, we can see that the graphs with the most variation are glucose, pregnancy, age, and glucose Pedigree Function. When we view the histograms, there are some features that are normally distributed. Most of the features are heavily skewed. This may lead me to believe the Gaussian NB classifier may not be the best choice, but it is a large data set with some continuous features.

For our model selection, we decide to modify the data to have holdout data be our testing data. This is the data that does not include a zero in the features Blood Pressure, BMI, and Glucose. This makes sense to test on the data that does not have missing values. Though, it may impair our training and in turn impair our testing as well. Here is our totals with the holdout testing data.

	Name	Score
0	KNN	0.729282
1	DT	0.729282
2	GNB	0.734807
3	BNB	0.657459

We can see from this data that Bernoulli NB will not be used in this perticular dataset. There are too many features with complex data for a binary disribution. We can also see that Decsion Trees and KNN have a similar accuracy. It is noted that Gaussian NB has the highest accuracy, but 0.73 is not an acurate classifier. We need to improve our data with cross-validation and finding optimal parameters.



After cross-validation seen on the right, we can see that KNN decreases along with Decision trees. In experiment 2, we see that decision trees are well suited for this data when the optimal parameters are found. I believe finding the correct number of parameters and finding the most relevant features will raise this accuracy. GNB will not be improved past this point and KNN will not be improved much with the optimal number of neighbors. We need to find the correct features to use and the correct parameter numbers for this dataset to get a higher accuracy. Decision trees will be able to improve through optimality.

Iris Data

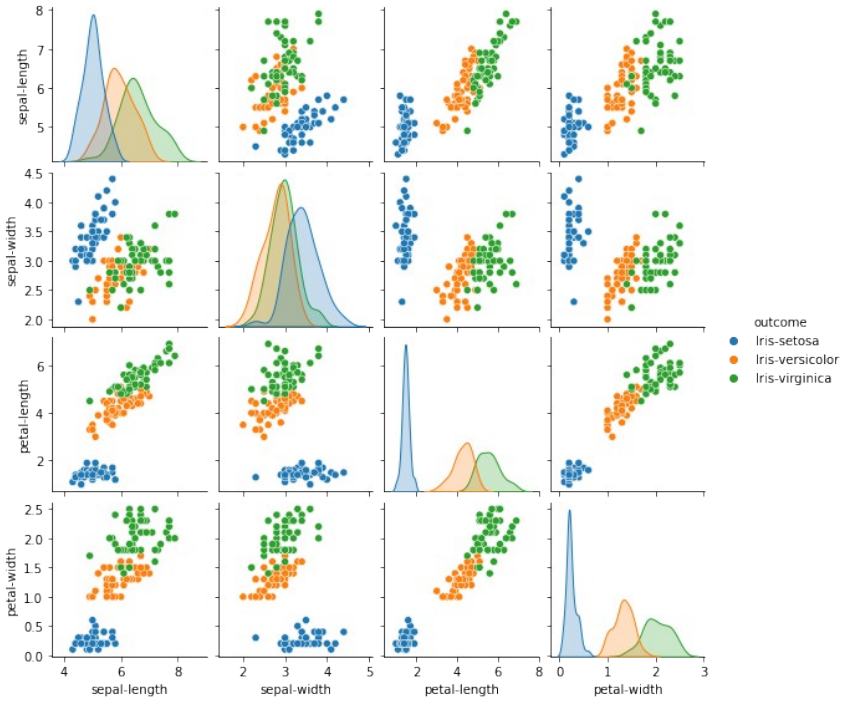
The Iris data has 150 subjects with 5 features. One of these features is the class. That feature is Outcome with three different outcomes: Iris-setosa, Iris-versicolor, and Iris-virginica. These are evenly distributed with 50 records in each. The other features are all numerical values that appear to be continuous at first glance. This distribution and numerical values makes me think of Gaussian NB for our classifier.

The averages of the data have different averages with relatively small standard distributions. This makes me think they are clustered. We do not appear to have any missing values with no minimums of 0 for any feature.

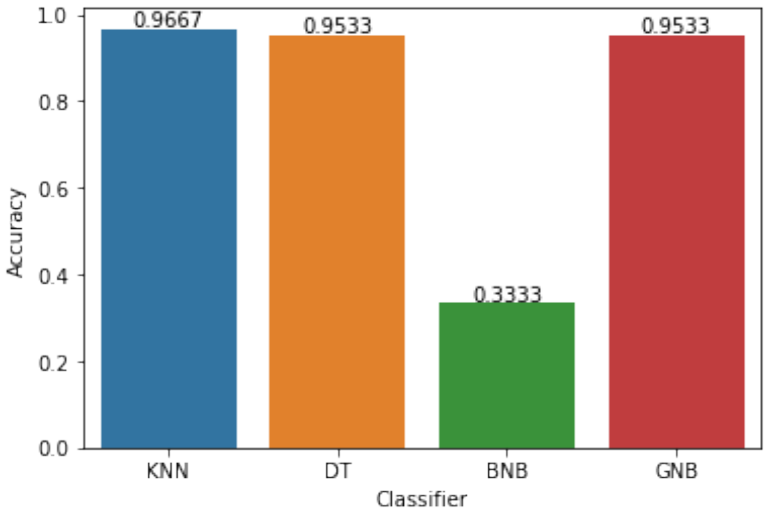
Through the plots we can see we were correct in our assumption that there are some clusters forming in the data. With a small dataset, I think we will not worry too much with runtime and be more concerned with accuracy. This data is appearing too small for Gaussian NB.

We can also see that petal length and petal width have discrete correlations due to the curve. It shows the three individual peaks with little overlap. We can clean the data to make it discrete so this dataset is ideal for decision trees or KNN.

With the little number of features, it is not necessary to try and find irrelevant features as none of them appear irrelevant in the plots.



With no missing values and no irrelevant features, we can use cross-validation to get a correct accuracy for our dataset.



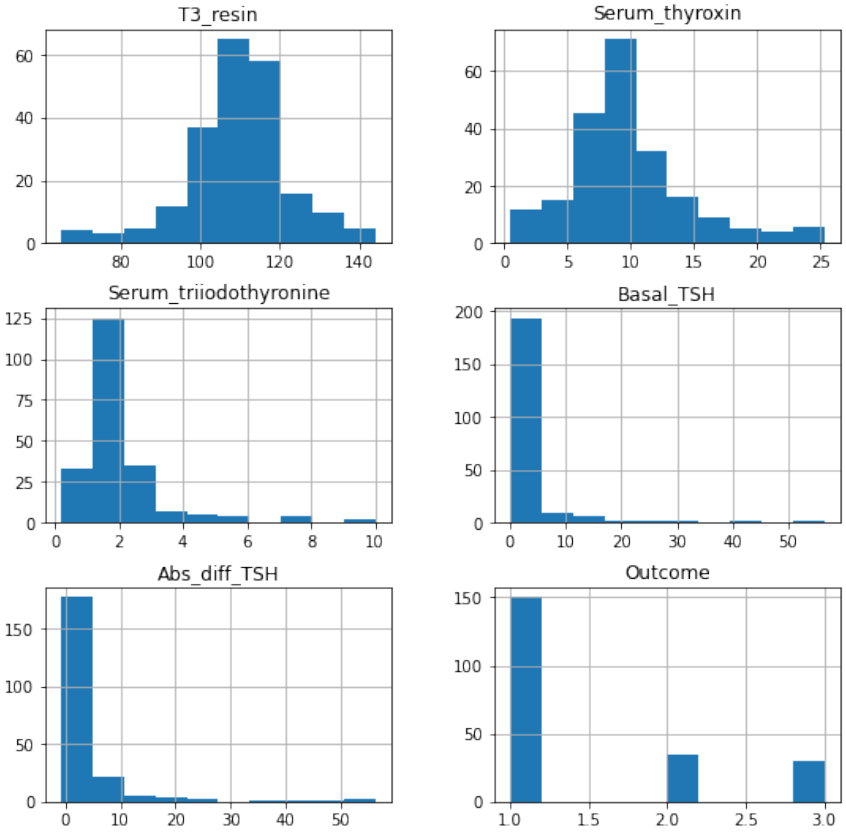
Through the cross-validation data, we can see the BMB is not the correct classifier for this data. We will not be concerned with such a low accuracy. GNB has been eliminated with the size of the dataset and the non-complex data. Decision trees have a great accuracy, but KNN will be the optimal choice. It has a high accuracy with this clustered data. Since the size of the dataset is low, we will not have to worry about using lots of power and time to use this classifier as well.

Thyriodism

We have 215 patients with 6 features. One of the features is the Outcome which will be our classes. The three classes are 1, 2, and 3. The classes are skewed towards 1 having 150 records. Classes 2 and 3 having 35 and 30 records respectively. All the other features are numerical data and appear to be continuous since they are float64 data type.

Viewing the histograms for this data shows us that these features seem to have a normal distribution. The data seem complex and the averages do not lead us to believe we will be able to use Bernoulli NB classifier. There are some features that appear to be skewed to the left, but the classes are skewed to the left as well.

There does not appear to be any null values which leads me to believe there are not missing values in this data. It is not showing any patterns or glaring discrepancies at first glance of the histograms.



The graphs show us there does not appear to be similar averages in these features. T3_resin is in the hundreds while the rest vary within two digits.

With no missing values and complex data, two choices for classifiers would be Decision Trees and Gaussian NB. They would be able to decipher this data and find where the classes lie. Decision Trees may have an overfitting issue with the large data size.

KNN does not appear to be an obvious choice in this dataset. The dataset is a little to large and the data seems to be normally distributed throughout.
We will use cross-validation and compare the accuracy of each classifier to see the difference.

Our assumptions have proved correct through the cross-validation results. KNN and DT have high accuracy but the data is perfectly fit for the Gaussian NB classifier. The normal distribution of the features and the size of the dataset make it a great fit for the GNB while the others fall short in the accuracy test.