A list of the top 10 bag of word features selected by filtering by frequency.

Top 10 words by frequency:

	Word	: Frequency
1.	'to'	: 1269
2.	'you'	: 921
3.	'I'	: 881
4.	'a'	: 854
5.	'the'	: 753
6.	'in'	: 567
7.	'and'	: 543
8.	'is'	: 524
9.	'i'	: 460
10.	'for'	: 437

A list of the top 10 bag of word features selected by filtering by mutual information.

Top 10 words by mutual information:

	Word	: Mutual Information
1.	'Call'	: 0.03707
2.	'to'	: 0.0296
3.	'call'	: 0.0280
4.	'or'	: 0.0270
5.	'FREE'	: 0.02696
6.	'claim'	: 0.0249
7.	'To'	: 0.0231
8.	'mobile'	: 0.0228
9.	'&'	: 0.0227
10.	'Txt'	: 0.0223

Code for your implementation of FindMostFrequentWords and FindTopWordsByMutualInformation (and supporting code)

See Attached Files

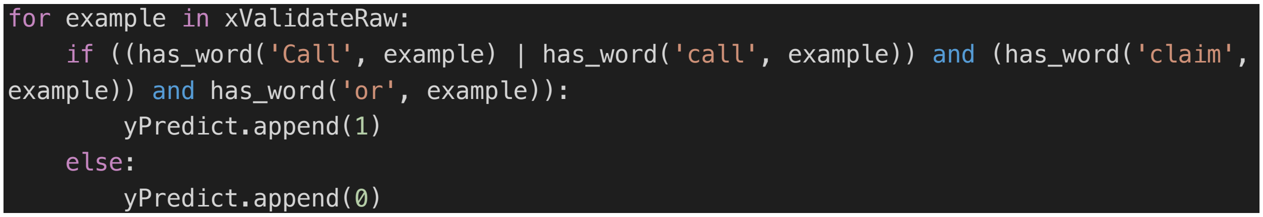
(Short answer 1-3 sentences) Compare the accuracy of the model learned with 10 most frequent words
to the model that predicts the most common class. Increase the number of features selected by 5 until you outperform
the most common class model. What number do you need?

The accuracy of the model learned with 10 most frequent words is the same as the accuracy from the most common class. To outperform the most common class model, we need top 25 most frequent words.

Run logistic regression with the top 10 words by mutual information.
Produce a table showing the selected words and the weights learned for them

Word	Weights
'Call'	0.814129
'to'	0.774774
'call'	0.842243
'or'	0.785383
'FREE'	0.524681
'claim'	0.466629
'To'	0.505128
'mobile'	0.505992
'&'	0.522082
'Txt'	0.482053

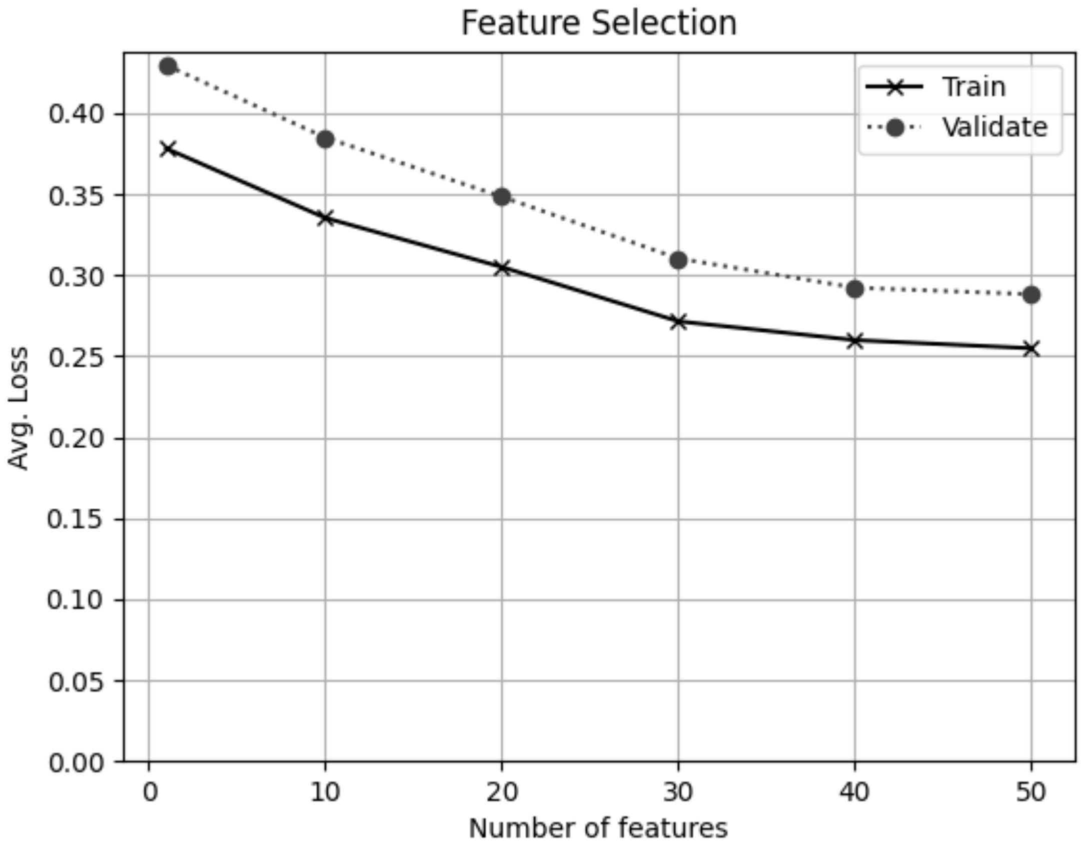
Create an if statement that partially matches the linear model, classifying some of the same messages
as (with no additional false positives). Use no more than 5 clauses in the if statement spam.



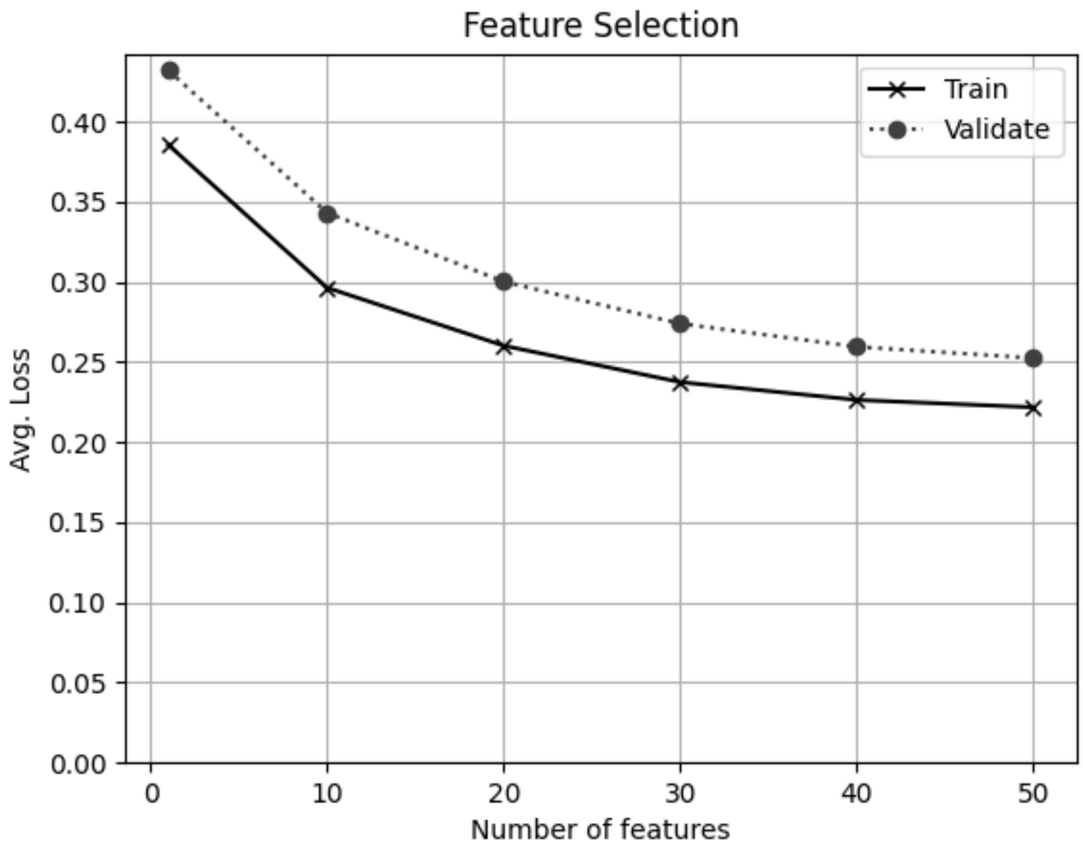
```
for example in xValidateRaw:
    if ((has_word('Call', example) | has_word('call', example)) and (has_word('claim', example)) and has_word('or', example)):
        yPredict.append(1)
    else:
        yPredict.append(0)
```

[redacted]

* Perform a parameter sweep on the number of features to use as selected by frequency, using n = [1, 10, 20, 30, 40, 50]
-- Produce a plot with the number of features used on the X axis, and the train and validation losses plotted on the y axis
-- Make sure to label the chart correctly and completely! (in the future you will lose 0.5 points for anychart that isn't properly labled)



* Perform a parameter sweep on the number of features to use as selected by mutual information, using n = [1, 10, 20, 30, 40, 50]
-- Produce a plot with the number of features used on the X axis, and the train and validation losses plotted on the y axis



* Provide short (1-3 sentence) answers to the following:
-- Which feature selection seems better based on the information you have? Why?

Selecting features by mutual information is better than selecting by top-n frequent words because of overall less log-loss in both the validation and training sets. It also appears that logistic regression uses less number of features using mutual information. A log loss of 0.30 is achieved with 10 features selected by mutual information, whereas we need 20 features if selecting by top-n frequent words.

-- Would it make sense to try n = 100 with mutual information based feature selection? Why?

It could make sense to try n = 100 if it is desired to achieve a lower false negative rate. However, that is most likely not the case in a spam detection model that strives to a achieve 0 False positive rate. We already achieve that with n = 50.