**NATIONAL UNIVERSITY OF SINGAPORE**

School of Computing

**IS5126** - Hands-on with Applied Analytics

Semester 2 – AY 2023/24

ML Guided Project

**Group Submission**

**SMS Spam or Ham**

**Group Members –**

Putri Darmawan (A0200717R)

Premi Jeevarathinam (A0268262Y)

Cai Zhi Hao (A0290917W)

**Date –**

14th February 2024

Table of Contents

[Question 2 3](#_Toc158817882)

[Linear classifier 3](#_Toc158817883)

[Majority-class classifier 3](#_Toc158817884)

[Question 3 4](#_Toc158817885)

[(a) Visualization of Logistic Regression Model on Unit Test Data 4](#_Toc158817886)

[(c) Hyperparameter Tuning for Convergence Parameter 4](#_Toc158817887)

[(d) Interpretation of Hyperparameter Sweep Results 4](#_Toc158817888)

[(e) Loss Analysis and Comparison 5](#_Toc158817889)

[Question 4 6](#_Toc158817890)

[(a) Code Implementation for top 10 words 6](#_Toc158817891)

[(b) Logistic Regression with Top 10 Words by Frequency 7](#_Toc158817892)

[(c) Parameter Sweep on Number of Features by Frequency 7](#_Toc158817893)

[(d) Parameter Sweep on Number of Features by Mutual Information 8](#_Toc158817894)

[(e) Comparison of Feature Selection Methods 8](#_Toc158817895)

[Question 7 9](#_Toc158817896)

[(a) Initial Hyperparameter Optimization Results 9](#_Toc158817897)

[(b) Final Validation Accuracy Improvement Plot 11](#_Toc158817898)

[(c) The Final Learned Hyperparameters 11](#_Toc158817899)

[(d) The ROC Plot: Best Model vs. Initial Model on Test Set 11](#_Toc158817900)

# Question 2

## Linear classifier

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Confusion Matrix | [[325, 135], [23, 64]] |
| Accuracy | 0.71 |
| Precision | 0.32 |
| Recall | 0.74 |
| False Positive Rate | 0.29 |
| False Negative Rate | 0.26 |

Low precision means when the model predicts a positive, it has high chances for it to be negative.

Precision =

## Majority-class classifier

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Confusion Matrix | [[460, 0], [87, 0]] |
| Accuracy | 0.84 |
| Precision | 0.00 |
| Recall | 0.00 |
| False Positive Rate | 0.00 |
| False Negative Rate | 1 |

This model will predict all as negative, since there is no TP, Precision, recall will be 0, and no FP means FPR will be 0.

# Question 3

## Visualization of Logistic Regression Model on Unit Test Data

1. iterations of gradient descent, stepSize=1.0, convergence=0.005

A colorful square with red dots

Description automatically generated

## Hyperparameter Tuning for Convergence Parameter

|  |  |  |
| --- | --- | --- |
| **Convergence params** | **Steps to convergence** | **Validation set accuracy** |
| 0.01 | 8 | 0.84 |
| 0.005 | 19 | 0.84 |
| 0.001 | 57 | 0.87 |
| 0.0001 | 186 | 0.92 |
| 0.00001 | 522 | 0.93 |

**Note** - smaller values of the convergence parameter led to more steps for convergence but may result in higher accuracy.

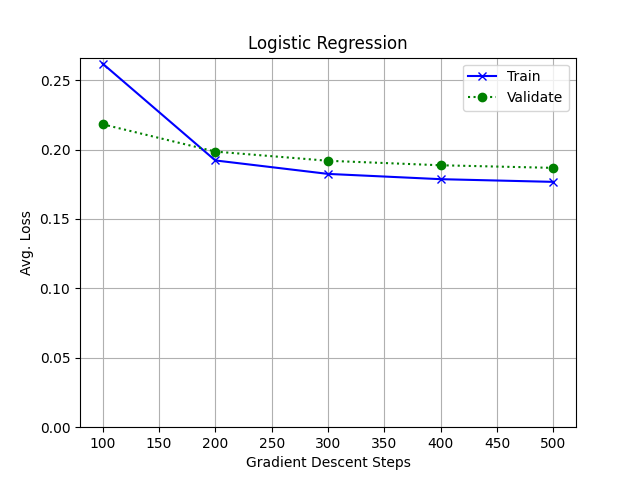
## Interpretation of Hyperparameter Sweep Results

* Smaller values of the convergence parameter led to more steps for convergence but may result in higher accuracy.
* The increase in accuracy with decreasing convergence parameters suggests the model is learning better (it learns until it cannot improve significantly).
* If further exploration were warranted, the next value to try might be 0.000001, as it represents another order of magnitude decrease in the convergence parameter. It is because, from the table, the accuracy is still increasing, we suspect that if we further decrease the convergence parameter, the accuracy will still increase.

**Output**:

|  |  |  |
| --- | --- | --- |
| **Convergence Parameter** | **Steps to Convergence** | **Validation Set Accuracy** |
| 0. 000001 | 1216 | 0.93 |

## Loss Analysis and Comparison



It seems that the Validation loss is worse than the Train loss in general. This might be because the model is able to learn the pattern in the training data well, but it might find it a bit hard to deal with unseen data (in this case validation data). The model performed better in training data compared to validation data because it has learned a lot from the training data, but it has not seen the validation data.

# Question 4

## Code Implementation for top 10 words

**FindMostFrequentWords**

def FindMostFrequentWords(self, x, n):

        # print("Stub FindMostFrequentWords in ", \_\_file\_\_)

        if n == 0:

            return []

        frequency = {}

        for data in x:

            tokens = list(set(self.Tokenize(data)))

            for token in tokens:

                if token not in frequency:

                    frequency[token] = 0

                frequency[token] += 1

        frequency\_items = frequency.items()

        frequency\_items = sorted(frequency\_items, key=lambda x: x[1], reverse=True)

        nth\_score = frequency\_items[n - 1][1]

        # handle ties

        return list(

            map(lambda x: x[0], filter(lambda y: y[1] >= nth\_score, frequency\_items))

        )

**FindTopWordsByMutualInformation**

def FindTopWordsByMutualInformation(self, x, y, n):

        # print("Stub FindTopWordsByMutualInformation in ", \_\_file\_\_)

        if n == 0:

            return []

        frequency = {}

        for data, label in zip(x, y):

            tokens = list(set(self.Tokenize(data)))

            for token in tokens:

                if token not in frequency:

                    frequency[token] = {0: 0, 1: 0}

                frequency[token][label] += 1

        mutual\_info = {}

        p\_y\_pos = (sum(y) + 1) / (len(y) + 2)

        p\_y\_neg = (len(y) - sum(y) + 1) / (len(y) + 2)

        # total\_words = sum(sum(list(frequency[x].values())) for x in frequency)

        # print(total\_words)

        for key in frequency:

            x\_pos = frequency[key][1]

            x\_neg = frequency[key][0]

            p\_xy\_pos = (x\_pos + 1) / (len(x) + 2)

            p\_xy\_neg = (x\_neg + 1) / (len(x) + 2)

            # p\_x\_pos = (x\_pos + 1) / (len(x) + 2)

            # p\_x\_neg = (x\_neg + 1) / (len(x) + 2)

            p\_x = (x\_pos + x\_neg + 1) / (len(x) + 2)

            mutual\_info\_pos = p\_xy\_pos \* math.log2(p\_xy\_pos / (p\_x \* p\_y\_pos))

            mutual\_info\_neg = p\_xy\_neg \* math.log2(p\_xy\_neg / (p\_x \* p\_y\_neg))

            mutual\_info[key] = mutual\_info\_neg + mutual\_info\_pos

        # print(mutual\_info)

        mutual\_info\_items = mutual\_info.items()

        mutual\_info\_items = sorted(mutual\_info\_items, key=lambda x: x[1], reverse=True)

        nth\_score = mutual\_info\_items[n - 1][1]

        return list(

            map(lambda x: x[0], filter(lambda y: y[1] >= nth\_score, mutual\_info\_items))

        )

**Top 10 Bag-of-Word Features by Frequency and Mutual Information**

* Top 10 words by frequency: ['to', 'you', 'I', 'a', 'the', 'in', 'and', 'is', 'i', 'for']
* Top 10 words by mutual information: ['call', 'txt', 'free', 'claim', 'mobile', 'reply', 'i', 'now!', 'to', '&']

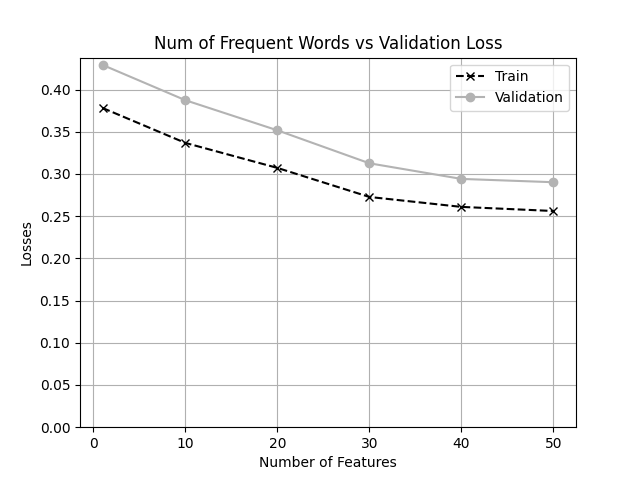
## Logistic Regression with Top 10 Words by Frequency

Logistic regression acc: 0.8409506398537477, most common class acc: 0.8409506398537477.

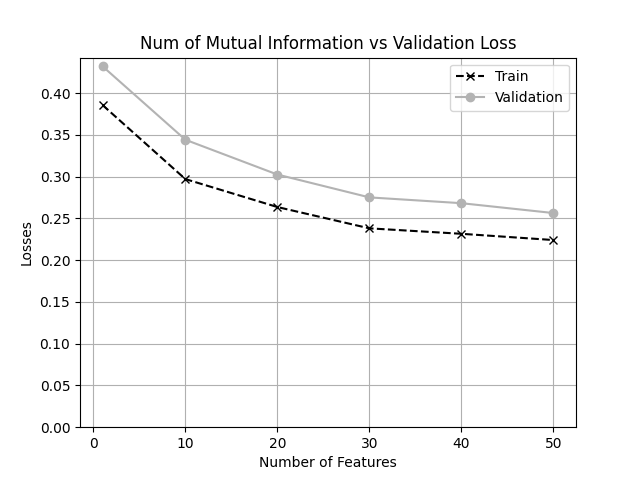
This means that it will only predict the data as negatives.

Using 25 frequent words will give better result than the most common class predictor, with accuracy 0.8574040219378428.

## Parameter Sweep on Number of Features by Frequency



## Parameter Sweep on Number of Features by Mutual Information



## Comparison of Feature Selection Methods

Mutual Information features are better than the number of top frequent words in general. As we can see from plot (c) and (d), the validation losses when using n top mutual information are lower than using n most frequent words. Mutual Information features work better because they highlight important information from the data. On the other hand, frequent words might include common words that don't really help with the task.

# Question 7

## Initial Hyperparameter Optimization Results

The order of hyperparameters tried:

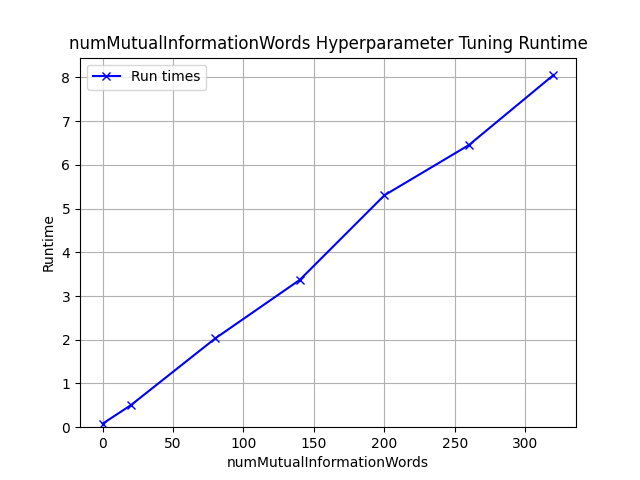
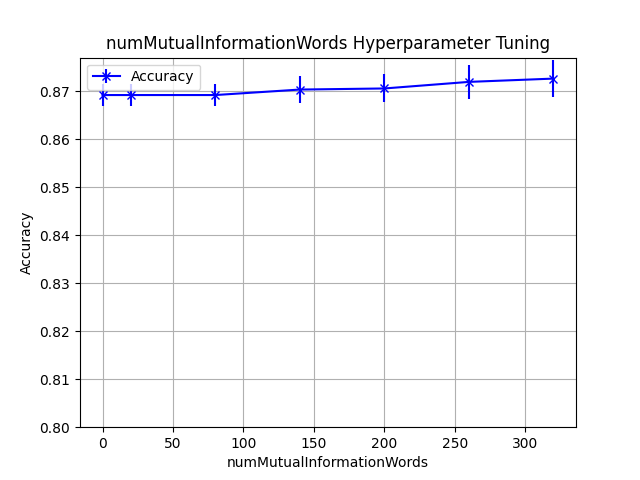
hyper\_params = { "numMutualInformationWords": [0, 20, 80, 140, 200, 260, 320],

"numFrequentWords": [0, 60, 120, 180, 240, 300, 360],

"stepSize": [0.05, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5],

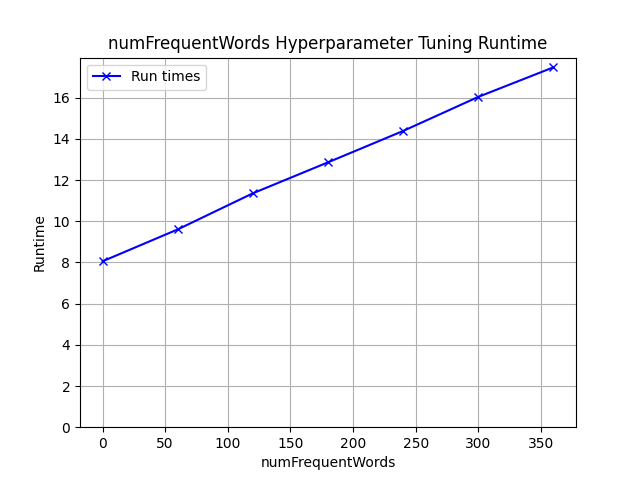
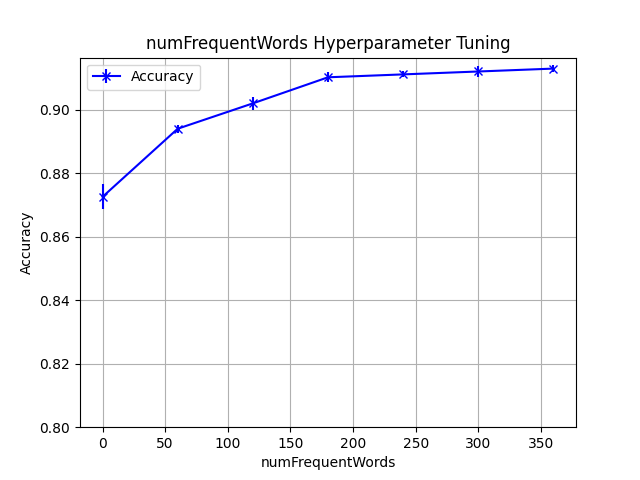
"convergence": [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05], }

**Mutual Information**



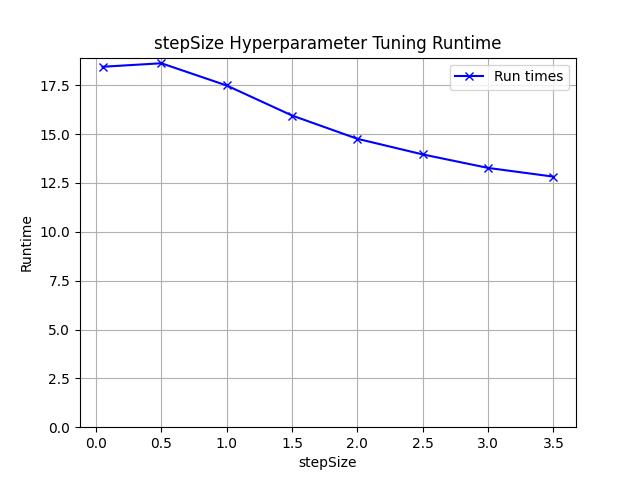
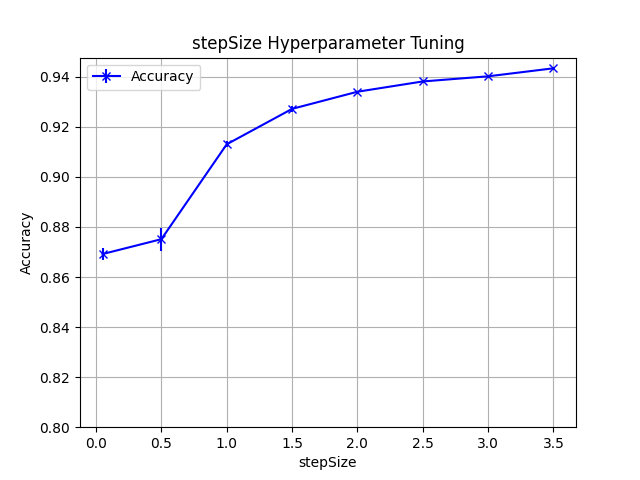
We think that we can further improve the accuracy by increasing the numMutualInformationWords since from the plot, the later part of the line keeps increasing. However, we will make the distance between each value in the sweep smaller to avoid overfitting.

**Frequent Words**



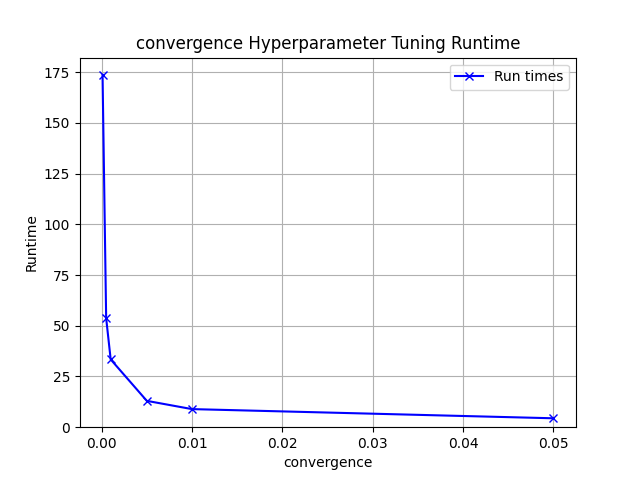
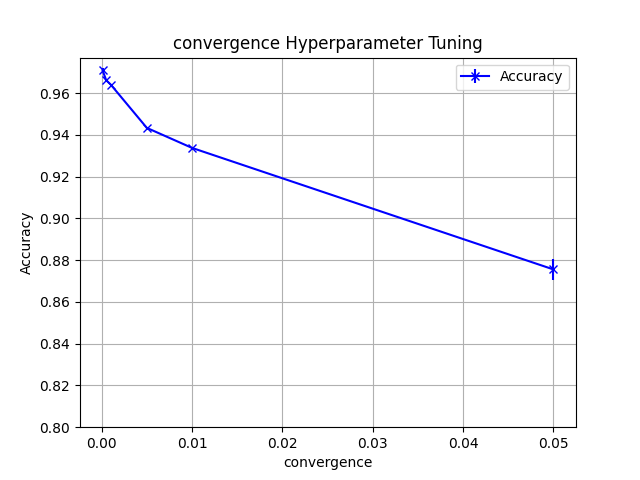
From the plot, it seems that as the number of frequent words becomes high, the increase is not that significant. We will not tune further since the difference of accuracy is not significant.

**Step size**



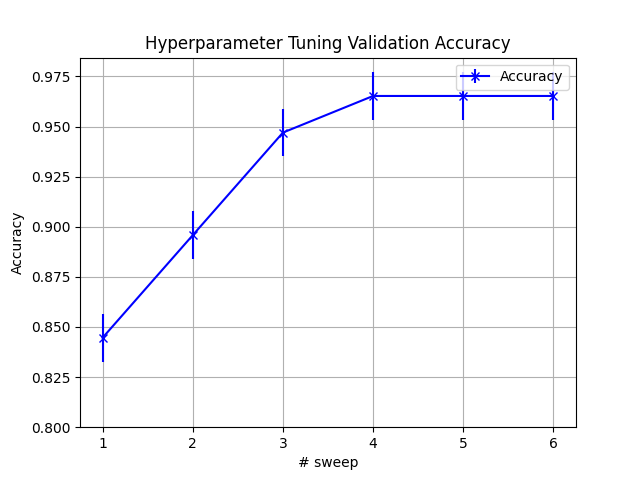
Higher step size means it can lead to faster convergence but might cause overshooting. However, in this case, we see that higher step size gives higher accuracy, thus we will try to tune further by increasing the step size.

**Convergence**



As we can see, the lower the convergence will give higher accuracy, but very slow run time. We will not tune further for this, as the runtime significantly becomes very slow.

## Final Validation Accuracy Improvement Plot



## The Final Learned Hyperparameters

best\_params = {'numMutualInformationWords': 360, 'stepSize': 4, 'convergence': 0.0001, 'numFrequentWords': 360}

## The ROC Plot: Best Model vs. Initial Model on Test Set

