# CS7641 Assignment 1 – Supervised Learning

Lirui Xiao (lxiao75)

Overleaf link: https://www.overleaf.com/read/hmpvmbjzsyjre2faa8

## 1 Dataset 1

### 1.1 Description

Email spam detection is a classification problem that holds significant relevance in today's digital communication landscape. Major email service providers such as Google, Microsoft, and Yahoo are continually developing and optimizing their email systems to filter out spam emails effectively. Each of these providers employs its own proprietary spam detection algorithms. The identification of spam emails is crucial for maintaining a clean and efficient communication channel. A high spam detection rate might lead to false positives, where legitimate emails are marked as spam, causing inconvenience to users. Conversely, a low detection rate fails to intercept spam emails effectively, leading to cluttered inboxes and potential security risks. Thus, achieving a balanced and accurate spam detection system is imperative.

I am particularly interested in this problem because it is closely related to our daily lives. Like many others, I have experienced the nuisance of spam emails and am keen to explore solutions to mitigate this issue. By engaging in this classification problem, I hope to contribute to improving email filtering systems and enhance user experience.

The dataset consists of over 5000 emails, with features extracted from the frequency of specific words within the email content. These features will serve as the basis for our classification model to determine whether an email is spam or not.

To address this problem, I will utilize three different kinds of algorithms as required by the task:

**Neural Networks**: For this spam detection task, I just use Multilayer Perceptron (MLP) to solve this issue.

**Support Vector Machines (SVM)**: For this problem, I experiment with two different kernel functions: the linear kernel and the radial basis function (RBF) kernel. The linear kernel is suitable for linearly separable data, while the RBF kernel can handle non-linear relationships by mapping the data into a higher-dimensional space. By comparing the performance of these kernels, I can determine which one is more effective for spam email classification.

**k-Nearest Neighbors (k-NN)**: In this task, we will experiment with different values of k (the number of neighbors to consider) to identify the optimal parameter. The k-NN algorithm's simplicity and interpretability make it a valuable benchmark against more complex models like neural networks and SVMs.

### 1.2 Hypothesis

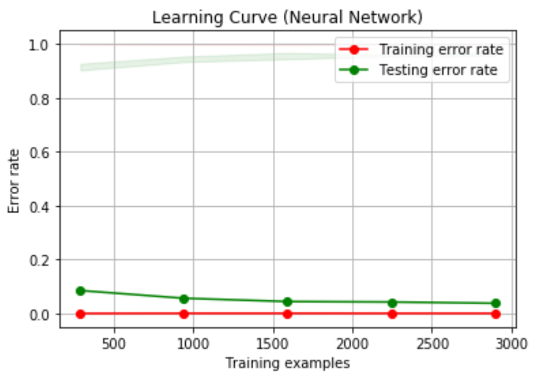
My hypothesis for this email spam detection dataset is that the frequency of certain keywords and phrases within an email is a strong predictor of whether the email is spam or not. Specifically, MLP neural network will outperform both Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN) in terms of classification accuracy and precision when applied to this dataset.

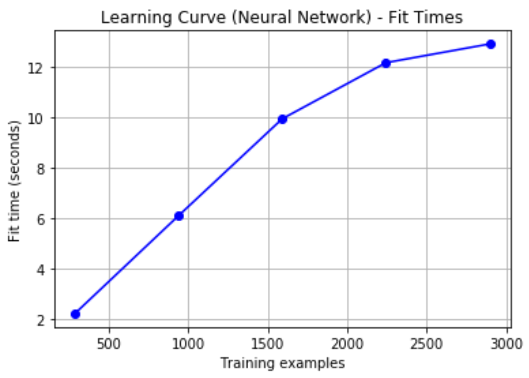
The rationales we assume is that MLP can automatically learn and extract relevant features from the data. This can be particularly advantageous in text data, where relationships between words and their frequencies might be complex and non-linear. SVMs can also handle non-linear decision boundaries effectively, but they might not capture complex interactions between features as effectively as neural networks. k-NN is easy to understand and implement.

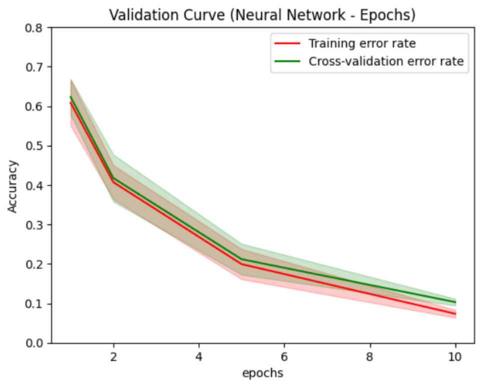
It works well with small datasets and provides a good baseline for comparison. However, it might struggle with large datasets and high-dimensional feature spaces due to the curse of dimensionality.

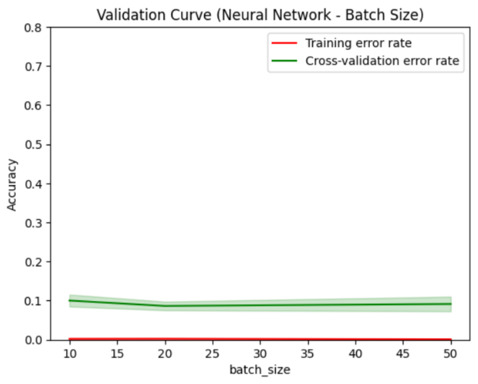
## 1.3 Training Results (Data and Graphs)

### 1.3.1 MLP (Neural Networks)

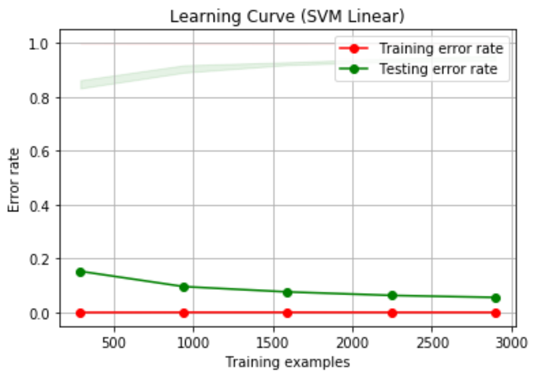


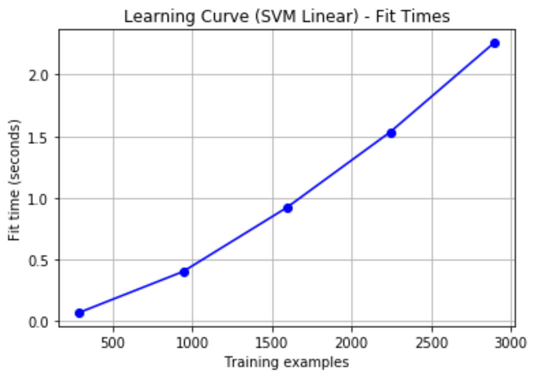


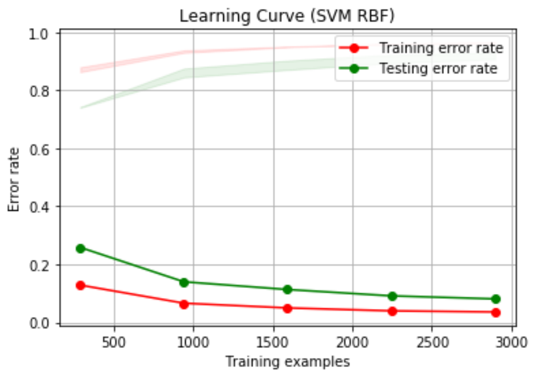


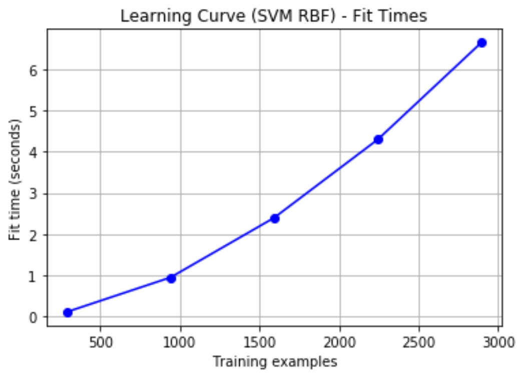


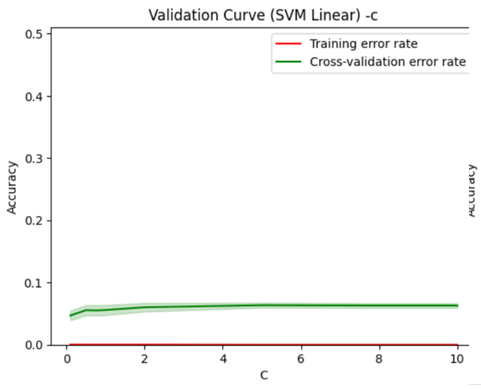
### 1.3.2 SVM

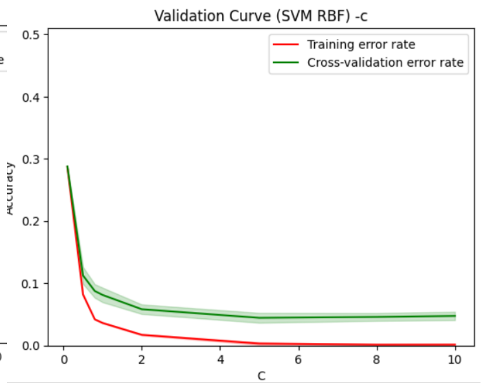


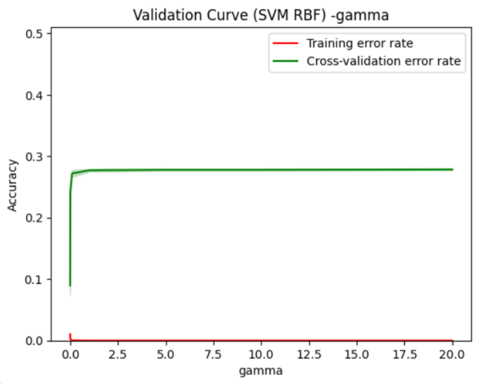




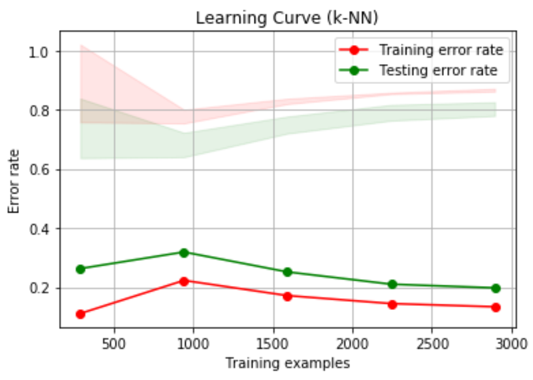


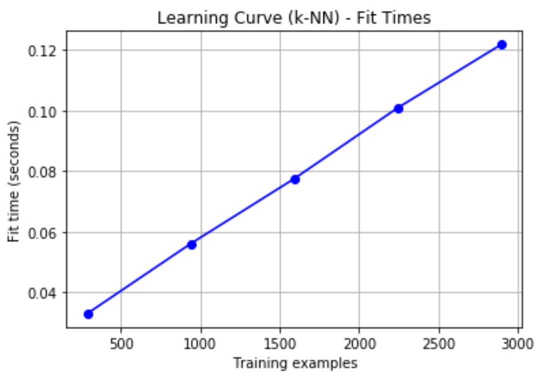


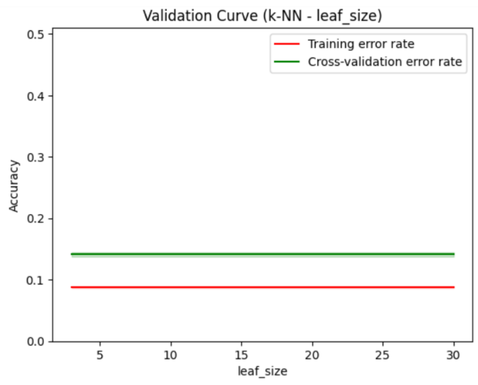


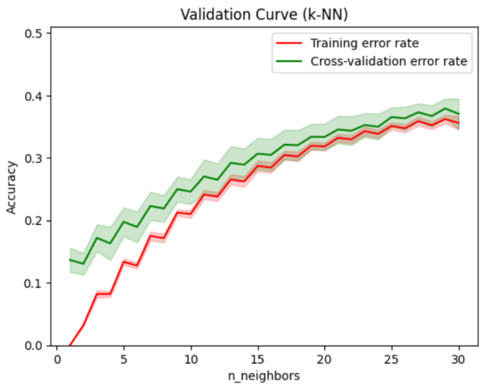


### 1.3.3 k-Nearest Neighbors









I progressively increased the training size to find the optimal size, and the test errors were observed from the stabilized points in the graph. We found that MLP has the lowest test error (0.04), followed by SVM with a linear kernel (0.06), SVM with an RBF kernel (0.07), and KNN (0.17). However, the advantage of MLP over the SVM algorithms is not significant. These results support our previous hypothesis.

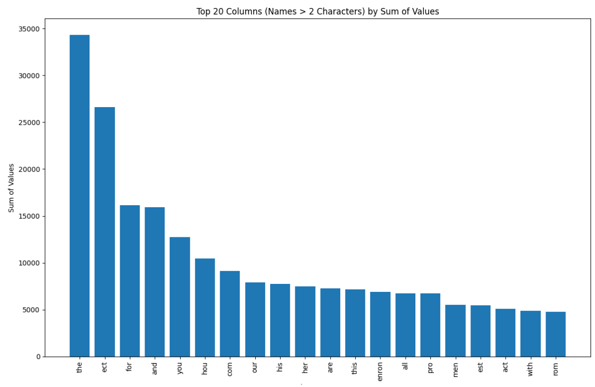
## 1.4 Analysis

To improve the performance of each algorithm, I used an approximate grid search to find the optimal parameters. For example, I tried the c and gamma parameter for SVM with RBF kernel, epoch and batch size for MLP model, leaf size and neighbor number for KNN. After their stabilization, I found every optimal parameter. In terms of computation complexity, among the algorithms I choose, KNN has lowest computation time. And the wall clock time for KNN and SVM linear increased linearly with the training size, whereas for MLP, the growth was a sub-linear exponential function.

During model evaluation, I used cross-validation instead of test error because cross-validation provides a more reliable estimate of model performance by averaging results over multiple folds, which reduces variance and gives a better indication of how the model will generalize to new data. The results show MLP has superior performance due to its ability to capture complex, non-linear relationships in the data. SVMs with an RBF kernel also perform well, but might not match the flexibility and feature learning capability of deep neural networks. k-NN provide reasonable baseline results, but may struggle with larger feature spaces and potentially higher computational costs.

The best algorithm I defined is a combination of cross-validation performance and computation time. This approach ensures that the selected model not only performs well on unseen data but also is computationally efficient, making it suitable for practical applications.

I loaded the dataset and performed preprocessing by removing common stop words, converting text to lowercase, and stripping punctuation. By summing the word counts and filtering for words with more than two characters, I identified the top 20 most frequent words. The results indicated that words like "act" "ect," and "com" are highly frequent, suggesting common risk in the spam emails.



## 2 Dataset 2

### 2.1 Description

Another dataset I am working on is from a mobile company. It conducted a market survey to collect the prices of many mobiles, which helps to price its newly produced mobile phones. This dataset comprises 2001 entries, each detailing various features of the mobile phones, such as RAM, memory, battery power, and Front Camera mega pixels. The aim of my project is to employ machine learning to identify the relationship between these features and the phone's price range (categorized from 0 to 3, where 0 represents the lowest price range and 3 represents the highest) and then make predictions for new data.

One interesting aspect of this dataset that inspires me is its practical purpose. By gaining a deeper understanding of how mobile phone features influence pricing, I can make more informed decisions when purchasing a phone in the future. It can help me avoid being misled by sales personnel and ensure that I am paying a fair price for the features that matter most to me.

To address this multiple class classification problem, I utilize three kinds of different algorithms like the above dataset. But some algorithms are different:

**Neural Networks**: For this mobile pricing range prediction, I design a neural network with multiple layers (deep learning) and experiment with various activation functions such as ReLU (Rectified Linear Unit) and SoftMax.

**Support Vector Machines (SVM)**: For this problem, I experiment with two different kernel functions: the modified version –SVC with the linear kernel and the radial basis function (RBF) kernel, which is very suitable to multiple class classification.

**k-Nearest Neighbors (k-NN)**: I used the same algorithms as in the previous dataset.

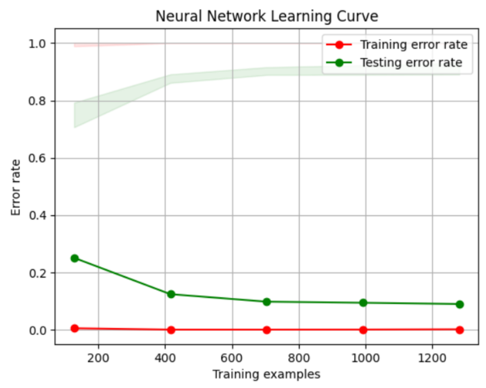
**Decision Tree Boosting (Extra credit)**: I use a boosting algorithm to enhance the performance of decision trees. Specifically, it creates a base decision tree classifier and uses the AdaBoost algorithm to boost it.

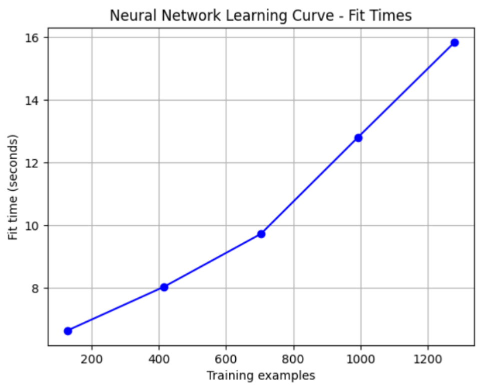
### 2.2 Hypothesis

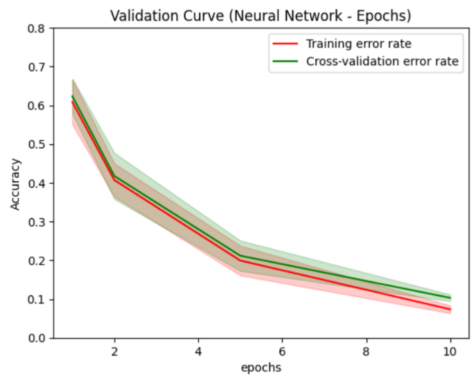
My hypothesis is that the most significant features influencing the price range of mobile phones in the dataset are RAM, memory, and network type. Specifically, I hypothesize that higher RAM and memory capacities, as well as support for newer network types like 5G, are strong indicators of a higher price range. Additionally, I propose that the type of SIM card (e.g., virtual vs. physical) has a less significant impact on the price range compared to the features. In terms of models, Neural network with multiple layers and activation functions will accurately predict the mobile pricing range. The SVM with an RBF kernel will outperform the SVM with a linear kernel in predicting the mobile pricing range. The boosting algorithm will enhance the performance of decision trees, resulting in better accuracy for mobile price range prediction. The rationale is that the flexibility in designing neural network architectures allows capturing intricate patterns, making them well-suited for complex, non-linear problems like price range prediction. Additionally, boosting algorithms like AdaBoost enhance decision tree performance by combining multiple weak learners into a stronger model, improving overall prediction accuracy.

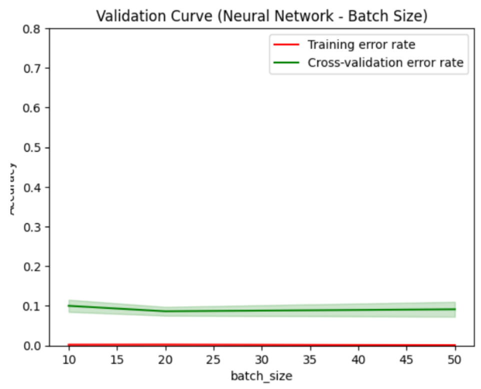
## 2.3 Training Results (Data and Graphs)

### 2.3.1 MLP (Neural Networks)

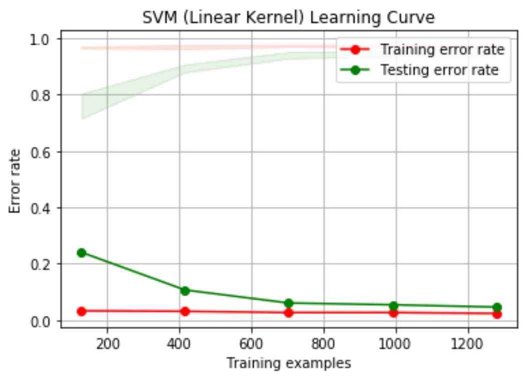


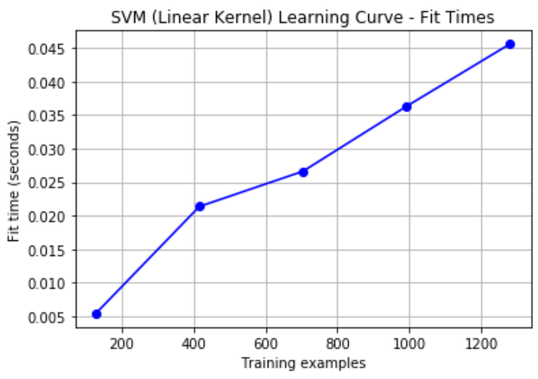


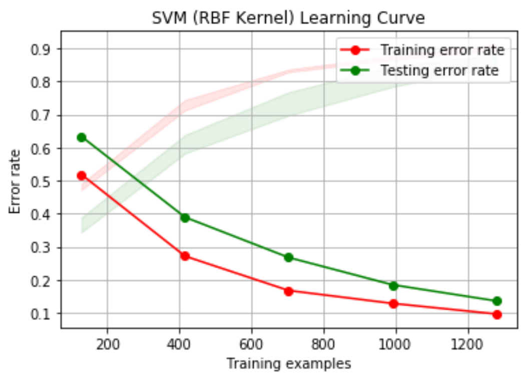


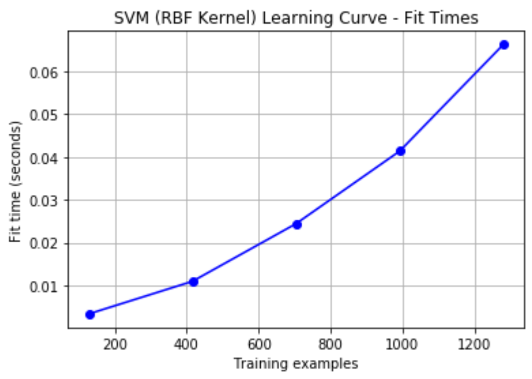


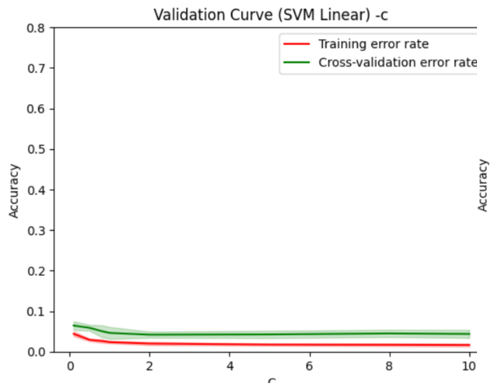
### 2.3.2 SVM

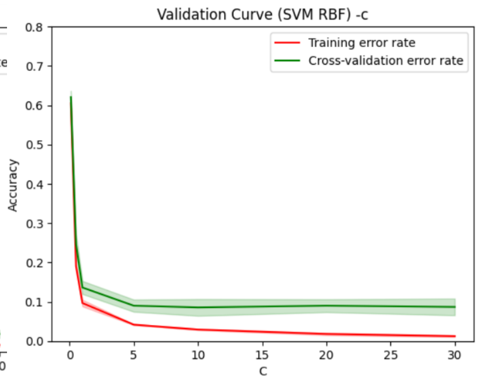


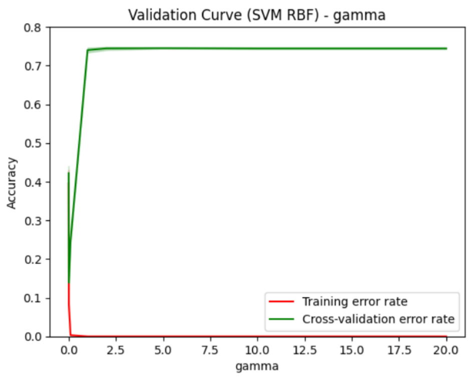




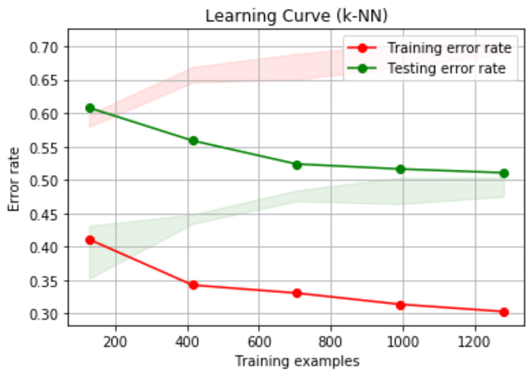


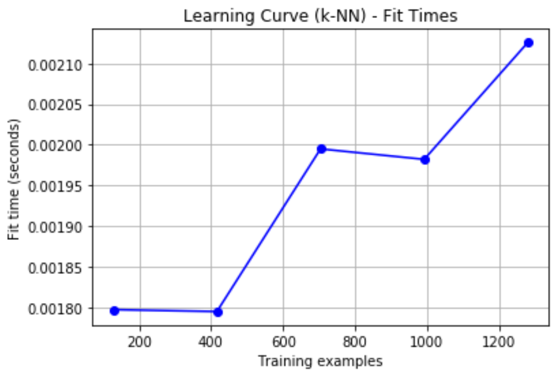


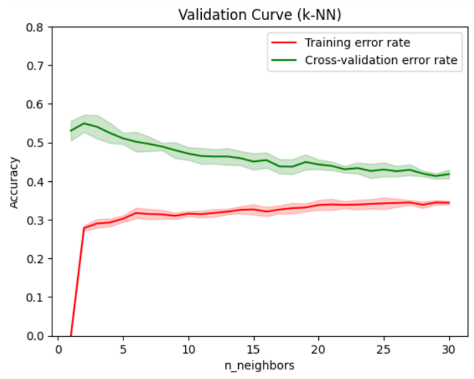


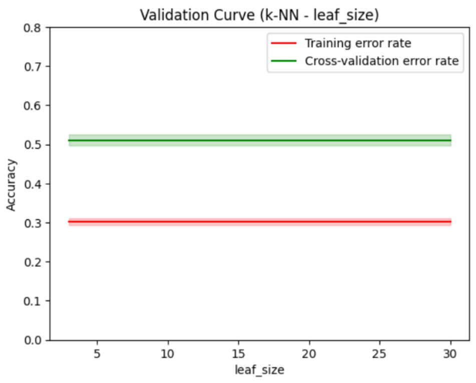


### 2.3.3 k-Nearest Neighbors





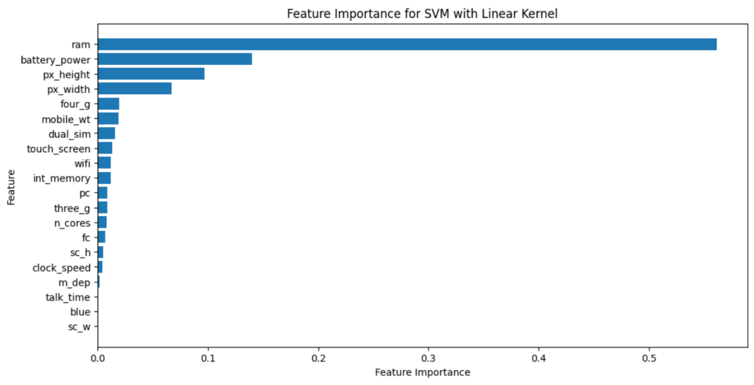




I progressively increased the training size to find the optimal size, and the test errors were observed from the stabilized points in the graph. I found that on this dataset, the SVM linear model performs the best, indicating that the data is linearly separable to a significant extent. Additionally, the neural network (NN) and SVM with RBF kernel perform moderately well, with test errors ranging between 0.1 and 0.2. Surprisingly, the KNN model exhibits a test error exceeding 0.5, suggesting that the dataset's feature space might be high-dimensional or that the relationships between features and the target variable are too complex for KNN to handle effectively. Overall, these results generally align with the hypothesis, though there are some discrepancies.

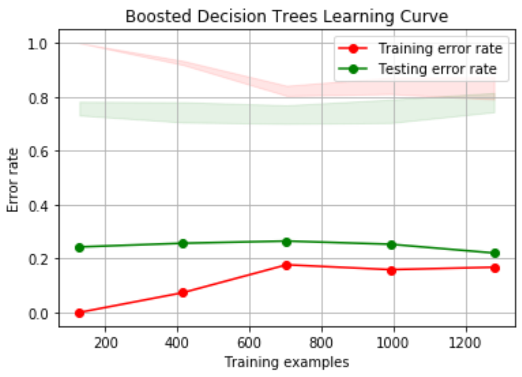
## 2.4 Analysis

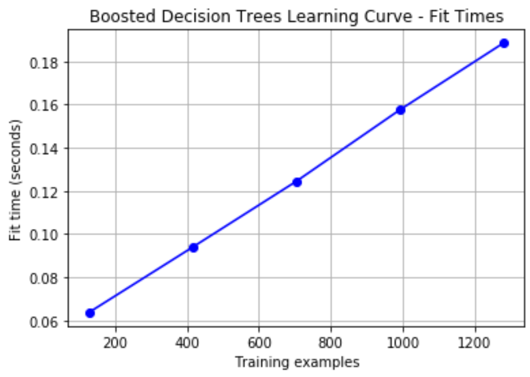
Like with the previous dataset, I used an approximate grid search to find the optimal parameters. In terms of computational complexity, the SVM with a linear kernel and KNN had the lowest computation time. The best algorithm, as I define it, combines cross-validation performance and computation time. For this dataset, the SVM with a linear kernel performed well in both cross-validation error and computation time, making it the best choice for this dataset. I used the best model to explore the feature weights (SVM with a linear kernel). I found that found that RAM is the most important feature, with a weight exceeding 0.5, which is greater than the combined weights of all other features. This aligns with our intuition, as RAM is a core component of a phone's performance and cost. Additionally, battery power (0.15) and pixel height (0.1) are also significant features. Battery power is crucial for the phone's longevity and user experience, while pixel height affects the display quality, which is a key aspect of the phone's overall appeal and functionality.

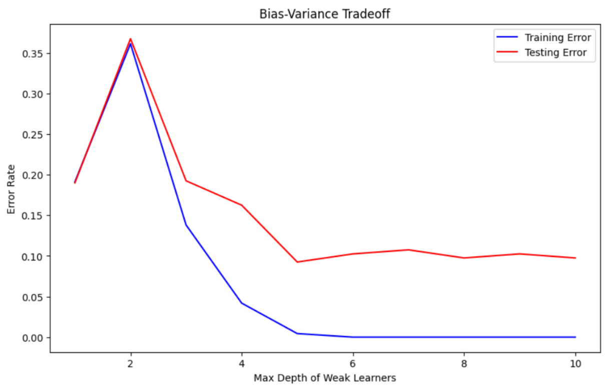


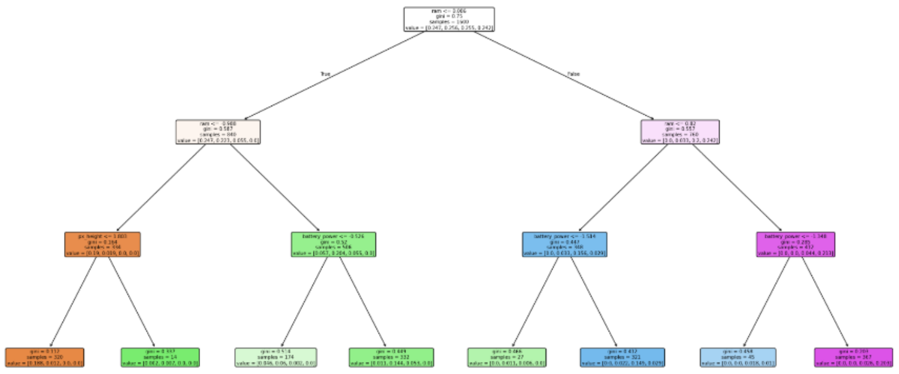
## 2.5 Decision Trees Boosting – Extra Credit Method for dataset 2

In terms of pruning methods, I used the max_depth=3, min_samples_split=20, and min_samples_leaf=5 parameters to limit the growth of the decision trees, thereby preventing overfitting and ensuring the trees remain weak learners. Weak learners have high bias and low variance because they are simple models that do not capture the complexity of the data well, leading to systematic errors. Boosting combines multiple weak learners to create a strong learner by sequentially focusing on the errors of the previous learners, which reduces bias as the model becomes more capable of capturing the underlying data patterns. However, boosting can increase variance because the combined model may become overly complex and sensitive to the training data, potentially leading to overfitting.









I use max_depth as an example. Above is a graph illustrating the bias-variance tradeoff in this boosting algorithm with varying maximum depths as weak learners. As the maximum depth increases from 2 to 5, both the training and testing errors initially decrease, indicating a reduction in bias. However, beyond a depth of 5, the training error continues to decrease while the testing error stabilizes and starts to increase slightly, demonstrating an increase in variance due to overfitting. This graph highlights the importance of selecting an optimal depth for weak learners. Not all parameters improve the model's performance by simply increasing their values.

I then examined the classification performance of the AdaBoost model. I observed that RAM is often used as the first split criterion in the decision trees, which aligns with our previous observations of feature importance. Additionally, battery power and pixel height are also frequently used as top split criteria. This finding is consistent with our intuition.