The focus of this project revolves around machine learning. Before we discuss the results, we are going to give a high level overview of the machine learning processes used in this project.

First order of business is describing what machine learning is. Machine learning is a kind of artificial intelligence that focuses largely on pattern recognition. The kind of machine learning used for this project is called classification. Before we get into the meat of what classification is, we want to leave you with a one sentence summary of what machine learning is. Machine learning is using the vast computational power of computers to find useful patterns within data that humans could not otherwise have found.

The kind of machine learning this project is concerned with is classification. Classification, as the name implies, deals with classifying things. The archetypical example is a spam filter. When an email comes into an inbox, modern email services will try and filter out the cases that are not relevant to you and put them in a spam folder. The underlying mechanism for how the program knows what is spam and what isn’t is machine learning classification. This process involves breaking down an email into features. Features are just ways that we describe an object. For this scenario, imagine that the features we are going to use are number of capital letters in the email, number of spelling mistakes in the email, and whether or not the email contains a hyperlink. Note that you could describe something like an email in a myriad of ways and the three we just listed were chosen arbitrarily. Envision now that we have thousands of labeled emails. By labeled emails, we mean emails that have *already been* classified as spam or not-spam. Can we then break the emails down into their features described above and find some sort of relationship between the values of the features and whether or not the email is labeled spam? Could we then give the model and new *unlabeled* email and, using all of the data we have already given it, have it accurately report whether or not the email is spam? Of course we can, and that is the backbone for machine learning classification.

Before we get into the actual algorithms used, we first need to have a brief discussion about the features we used in this project. For this project, we had to make features out of job descriptions that people gave, which are just free text. To do this, we used a *bag of words* model. Bag of words simply views text as the words used in a document and how often the words are used.

ADD PICTURE HERE.

That means when we are dealing with breaking down a new job description into its features, the features consist of the entire vocabulary, or every word that we have ever seen and the values for this features depend on how often those words show up in a given document. This is indeed an oversimplified representation of text, but it provides an intuitive and surprisingly robust way to perform text classification.

Using the text classification scheme just described, we can now talk about the algorithms used. This project implemented three different machine learning algorithms: Naïve Bayes, Support Vector Machines, and k-Nearest Neighbors. Naïve Bayes will be the only algorithm that will not be discussed in length as in involves knowledge of probability theory.

K Nearest Neighbors was the second algorithm that we used. To understand kNN, imagine that using the spam filter example above, we decided to only have two features for each email, say, number of capital letters and number of hyperlinks. For each instance, plot these values on a 2 dimensional graph shown below and label each point with whether or not that point corresponds to a spam or a regular email. Do this plotting for all the data that you want to use. Now, if someone gives you a new, unlabeled email, extract its features and plot them on the graph with all of the other. Now, look at this new points *k* closest neighbors, where *k* is a positive non-zero integers (usually odd), and whatever the majority class is for these *k* closest neighbors is what class you assign the new point.

ADD PICTURE HERE

The principle in higher dimensions is identical. Within this algorithm, one can tweak how large *k* is and what the distance metric between points. For this project, *k* was chosen using the algorithm where *x* is the number of labeled data points we are using. The distance metric used was squared Euclidean, given by but a more traditional measure is the cosine distance, which looks at the angle between points to determine the distance.

The final algorithm is support vector machines (SVMs). SVMs can be a bit mathematically rigorous, so we are only going to give a high level overview. Use the same example that we did in the previous paragraph where we have a bunch of labeled points on a 2-dimensional graph. An SVM is going to try and draw a line to separate the data classes, have all spam emails on one side of the line and all legitimate emails on the other. If it cannot perfectly separate them, it will do the best that it can. “Best as it can” is obviously a vague and subjective statement and this is intentional as the designer of the model gets to decide what the “best it can” is. After this line is drawn, any new instance is classified based on what side of the line it is on. It is possible and often advantageous to use separators other than a line, but this project stuck with only using a line.

ADD PICTURE HERE

The last topic that we need to cover here is how to train and test a machine learning model. This process consists of taking a large set of labeled instances and splitting that data set up into two disjoint group. The first group is called the train set and usually consists of 80% of the original data. As the name implies, this set of data is used to train the models. After the models are trained, we used the remaining 20% of the data to test the models and verify their accuracy. This set of 20% us called the test set. This consists of taking each data point within the test set and plugging each of the features of that data point into the trained model. The model will output a prediction based on the information is acquired during training. Compare this prediction to the actual label of the point and this is the foundation for validating a model. We used four different performance metrics to analyze the models. The first is *precision.* Using our spam example, precision becomes, whenever the model predicted that an instance was spam, how often was it right. Next is *recall*. Recall will sound a lot like precision, but there is a nuanced difference. Recall is, when the model was given an example of spam, how often did it predict it as such. *Specificity* is how the model predicted non-spam, how often was it right. Finally, we have *F-score.* F-score is a weighted average of the precision and recall and is a good overall metric for the performance of a model. In other words, if you had to stick one number on how well a model performs, F-score is a common choice.