Parsa Yadollahi - 260869949
COMP 551 Midterm

Section A:

Question 1:

1.1) Nearest Neighbours

The nearest neighbor algorithm solves both classification and regression problems and fits non-linear data. It works by calculating the distance between all points of a given dataset. This distance can be calculated using any distance method (Euclidean, Manhattan, Minkowski, etc). We then sort those distances in ascending order. The hyperparameter for this model is *k* and this determines how many points around the given value we consider the distance from. When the *k* is small we have a low bias, high variance, and tend to overfit, whereas when it is a bigger number we have a lower bias and tend to underfit. The higher our *k* value, the more computation time increase. KNN classification will look at the k-nearest neighbors we are trying to predict and will output the most frequent label among those *k* selected. KNN regression will output a numerical value representing the average of the labels of those *k* points.

1.2) Decision Tree

The decisions tree algorithm solves both classification and regression problems. The goal of the algorithm is to create a model that can predict the class of a target variable by learning decision rules derived from the training samples. We then iterate through each unused attribute to calculate its entropy and information gain on each iteration and select the attributes that contribute the least to entropy and most to IG and build a decision tree. We use the tree which provides the largest information gain. The algorithm tends to overfit so we can specify the maximum depth of the decision tree to avoid overfitting but this can lead to having a high bias. We use different cost functions to split the nodes between the Decision tree, entropy, Gini Index, and misclassification.

1.3) Naive Bayes

Naive Bayes models have different implementations (Gaussian, Multi classification, Bernoulli, etc) where some use class prior and others use likelihood. The model solves classification, more generally used for text classification. The classifier has a strong independence assumption between the features, hence the name Naive. The model is a conditional probability model meaning it calculates the conditional probability of a problem instance belonging to a particular class. Naive Bayes tends to underfit due to its strong assumption on independence between features, thus the bias is less flexible hence less likely to overfit.

1.4) Linear Regression

Linear regression is an algorithm that solves regression problems and takes a linear approach using the formula $y = mX + c$, where m is the slope of the line and c is the intercept, to model relationships between dependent and independent variables. We determine the relationship between these two variables by fitting a linear equation to the observed data. That fitting uses the least square approach by minimizing the cost function. The model takes the following form: yi = w0+w1x1+...+wpxp where wi represents the model parameters. If the model has high variance then the model tends to overfit.

1.5) Logistic Regression

Similar to linear regression, this algorithm uses the logistic function to model a binary dependent variable. We also now use maximum likelihood instead of the least square to fit our data and to find the best coefficients. Logistic regression can solve both regression and classification problems but is used mostly for regression to output if a given data entry belongs to either class 0 or 1. It can become a classification technique only when a decision threshold to separate classes. It works with both discrete and continuous data. Logistic Regression tends to overfit when there are high-dimensional settings or when the inputs are correlated. We can use the one-hot encoding method on this model.

1.6) Softmax Regression

Softmax regression also referred to as Multi-class Logistic classification, is a generalized version of the logistic regression that can be used for multi-class classification. To implement the cost function, we need to use the softmax cross entropy cost function which is the negative log-likelihood. Similarly to Logistic Regression, we can use one-hot encoding to represent our model labels with 0 or 1. And similarly to Linear Regression, we can use gradient descent to optimize our cost.

1.7) Multilayer Perceptron

Multilayer perceptron can be used for both regression and classification. The data must be numerical meaning if we have categorical data, we need to convert it using one-hot encoding. For regression, there is no activation function, the loss function is the Gaussian likelihood. For classification, the activation function is the logistic sigmoid, and the loss function is calculated using the Bernoulli likelihood. The middle layer consists of different activation functions (identity, hyperbolic tangent, logistic, ReLu, etc). MLP tends to overfit.

1.8) CNN

Convolutional Neural Networks is a neural network with convolutional layers. It's mostly used to analyze image data to for example classify images. Training is done through back propagation through convolution. The activation function is ReLu.

Question 2:

2.1) Linear Regression vs. Logistic Regression

Linear regression uses a linear plot whereas logistic regression uses the logistic function. Linear regression is used for predicting continuous dependent values whereas logistic regression is used for binary classification. Logistic regression uses maximum likelihood whereas linear regression uses the least square.

2.2) Logistic Regression vs. Softmax regression

Softmax Regression is a generalized version of Logistic Regression. It's a multiclass Logistic Regression that classifies a *k*-dimensional vector of values to k-dimensional labels bounded in the range (0,1). In Logistic Regression, we assume that the labels are binary. Logistic Regression uses maximum likelihood for the cost function whereas we use the negative log-likelihood for softmax regression.

2.3) Logistic Regression vs. Naive Bayes Classifier

Both models can be used for classification. Both models learn the following way: Naive Bayes estimates the join probability of training data. Whereas Logistic Regression estimates the probability from the training data minimizing error. In other words, Logistic Regression is a discriminative model whereas Naive Bayes is a generative model.

2.4) Logistic Regression vs. MLP

A single-layer perceptron using sigmoid activation is no different from Logistic Regression. As the number of nodes and layers increase in the perceptron, the model acts as a combination of multiple Logistic Regression models stacked together. This model predicts the class labels. In Logistic Regression, we use gradient descent to optimize the cost function whereas MLP uses backpropagation.

2..5) MLP vs CNN

Both MLP and CNN can be used for image classification. But their inputs are different. MLP takes vectors as inputs whereas CNN does not. CNN has a layer of convolution and pooling.

Question 3:

3.1) Over-fitting and Under-fitting

Over-fitting is a modeling error that happens when a function is too closely aligned to a limited set of data points. As a consequence, the model that overfits will likely not do well on other datasets.

Under-fitting is when a model is unable to capture the relationship between input and output variables. This is the opposite of over-fitting where it performs poorly on any dataset.

3.2) Bias and Variance trade-off

Bias is when an algorithm produces an incorrect result that is skewed due to an assumption that the model makes in the learning process. Variance is the amount a model will change when training on different datasets. Bias and Variance are complements of each other, thus the increase of one will result in the decrease of another. Meaning we are tasked with finding the right balance between the two. We ideally want a low bias and low variance.

3.3) Regularization

Regularization is a technique that modifies the learning algorithm. This modification will help the model generalize better by minimizing the loss function to avoid overfitting or underfitting.

3.4) Generalization

Generalization demonstrates how well a model that's been trained adapts when classifying unseen data chosen from the same distribution. In other words, it measures the performance of a model to process new data and generate accurate predictions after being trained.

3.4) Hyper-parameter

Hyper-parameter is a parameter whose value is chosen before a model is trained. It controls the learning process and does not affect the performance of the model. It affects the speed of the model learning and how well it does.

Question 4)

Gradient descent is an optimization algorithm that adopts an iterative process to find the local minimum of a function which allows us to optimize an algorithm. We essentially use it to minimize a cost function. Adam increases the speed at which gradient descent converges to the local minimum. It uses the square gradients to scale the learning rate and the moving average of the gradient.

Question 5)

<u>Section B:</u>

Question 6)

No gradient descent is not always guaranteed to find a local minima. It's simply an iterative method that follows the direction of the gradient. If sequence the gradient descent creates has an accumulation point, and that point has gradient zero, then it will stay stuck there. This is because the vector of all partial derivatives is 0.

Question 7)

Yes, we can use gradient descent in Linear Regression. Linear regression fits its data with a linear plot using ($y = mX + c$) where m is the slope of the line and c is the intercept, and its cost function is the least square. Our goal is to minimize the error and predict the parameters in the LR function. Gradient descent is an algorithm to find the minimum of a function and in this case, that function is the loss function. We can apply this algorithm to the linear regressions function parameters (m and c) to optimize it by calculating the partial derivative w.r.t m and updating their values until we converge to a local minimum or when our loss function is very small.

Question 8)

8.1) decrease variance, same bias
8.2) decrease variance, increase bias
8.3) decrease variance, same bias
8.4) decrease variance, increase bias
8.5) decrease variance, increase bias
8.6) increase variance, decrease bias

Question 9)

Regularization is more important for a model that has high expressiveness. A model that is more expressive means a model that overfits its dataset and will have a lower test error. We use regularization to reduce the variance and increase the bias.

Question 10)

Section C

Section D

Question 19)

I would use lasso regression because of its feature selection. When using lasso, we can set the coefficients to 0 meaning we can completely remove those features whereas this is not the case for Ridge regression. This works well with our scenario since we do not know what will impact the crops so we can remove those features that would not affect the model.

Question 20)

Recall measures the proportion of actually positive labels correctly identified by the model. In this case, the percentage of emails flagged as spam (positive).

If we were to increase recall, our model would try to classify as many true-positive labels which would increase the number of emails being sent to spam. But the downside of this is that it will likely increase the number of actual non-spam emails being sent to spam since it increases the number of false-positive. Thus, increasing the recall will increase, I expect to see less spam in my inbox and more actual emails being forwarded to spam.

Question 21)

21.1)

The biggest issue is that the dataset does not have many instances. We want to ensure that our model is well trained and has some instances to be tested on. To do so we should train the model with the most amount of instances. I would split my data roughly 80% training and 20% testing. I would not have a validation set but would instead opt to use k-fold cross-validation to train the hyper-parameter.

21.2)

Having a high training loss means that the model is performing badly. I would first start by determining if the data is properly being labeled, if this is fine I would move on to verifying if my hyper-parameter is the issue by tuning it again. If that is not the issue, I would move on to changing/modifying the model to see if it improves the result. I would then ask for more data. This could help our model a lot, especially

because the data we started off with was very little which could cause performance issues.

21.3)

If the model is returning a low training loss and a high validation loss, it likely means that it's gone past the turning point where the model is starting to overfit and can't generalize anymore. To solve this, I would check if the training and validation set have the same label distribution and shuffle the data. This could result in less variance between different runs.

21.4)

In normal circumstances, low validation and low testing loss are good since that's what we aim for. But in our case, there are a lot of external factors that could mislead these losses. This is because there is still a lot unknown about the small dataset, for example, if it is unbalanced, then a low loss would be justified. An example of this is if the entire dataset is true and your model predicts true 90% of the time. In the end, there should be more factors to consider like accuracy/recall if your model is ready to be deployed.

Question 22)

22.1)

When dealing with datasets that are small like this one having 1000 instances we want to select a model that will avoid overfitting as much as possible. This means we would need to make some assumptions about the distribution of the data. Some possibilities for this are Naive Bayes, Logistic Regression, or Decision tree. Now we notice that there are at least 3 classes listed, meaning, which rule out logistic regression and decision trees since Naive Bayes is more suited for multiclass classification.

22.2)