Predicting Breast Cancer Survivability Using Deep Learning and Support Vector Machines

Marian Qian, Computer Systems Research Lab, TJHSST
Dr. Patrick White, Lab Director

Abstract

This project evaluates whether combining a feedforward neural network and a radial basis function or linear support vector machine will help the prediction of breast cancer survivability. Breast cancer survivability is how long the patient survives after their initial diagnosis, and understanding survivability is crucial to aiding healthcare physicians in providing both effective and efficient treatment plans to patients. Previous papers have addressed this problem by using machine learning methods such as neural networks and support vector machines as individual algorithms, but this project investigates whether combining these two methods will improve the accuracy of survivability prediction. While the results of this project did not show benefits or an increased accuracy from combining a support vector machine with a neural network, a smaller standard deviation of model performances was shown in the radial basis support vector machine combined with neural network.

Introduction

Breast cancer is one of the most common cancers, not only in the United States but across the globe. Over 250,000 women in the US are affected by breast cancer each year (Center for Disease Control and Prevention [CDC], 2020), and by 2025, the expected number of cases will rise to approximately 19.3 million (Ragab et al., 2019). The prognosis of breast cancer, which describes the behavior of the disease after the initial diagnosis, mainly consists of recurrence, whether cancer will come back, and survivability, how long the patient will remain alive after the initial diagnosis.

Understanding these factors of prognosis would help guide treatment plans, such as whether a patient needs chemotherapy or hormone therapy, and proper follow-up methods that will most likely benefit the patient's survival rate of cancer (Boeri et al., 2020). While these treatments are crucial for reducing the spread of malignant cancer, over 70% of patients that receive these treatments would have survived without them (Park et al., 2013). Knowing patient survivability would allow patients to avoid unnecessary treatments and surgeries, such as biopsies, and

reduce hospital costs (Park et al., 2013), so machine learning methods have been used to predict these survivability outcomes that result in high accuracies (Boeri et al., 2020). My project aims to combine a deep learning algorithm, a feed-word network, and a radial basis function (RBF) and linear support vector machine (SVMs) algorithm on this same problem of predicting breast cancer survivability to see if this method can improve the accuracy of patient survivability.

Background

Machine learning methods, such as artificial neural networks (ANNs) and support vector machines, can find hidden patterns and relationships within data that would be difficult for humans to find (Kalafi et al., 2019) and therefore have a great potential in successfully predicting survivability. ANNs act similar to neurons in the brain: the algorithm consists of individual nodes that send signals which together determine the final output value. SVMs are an algorithm that separates predicted values using a hyperplane consisting of multiple dimensions (Boeri et al., 2020). Specifically for predicting breast cancer survivability, a 5-year threshold has been labeled to indicate whether patients survived cancer or not. If they lived longer than 5 years or 60 months, then they count as having survived the cancer (Kalafi et al., 2019). This is what the machine learning algorithms are predicting and what I will use for my project to determine which patients are in which classes.

Previous studies have investigated the use of machine learning methods, including support vector machines, decision trees, and random forests, to predict breast cancer survivability. The following two studies used data from the Surveillance Epidemiology, and End Results (SEER) dataset by the National Institute of Health that had a catalog of different cases of cancer tumors from various years. I also used this specific dataset for my project and based the variables and features I used for my model on these papers. Park et al. compared the performance of ANNs, SVMs, and semi-supervised learning (SSL) on breast cancer survival using the 5-year threshold and the SEER database. They found SSL performed the best, with 71% accuracy, then ANNs, with 65%, and then SVMs, with 51% accuracy. They used five-fold cross validation, which meant that they trained five separate models on five separate sections of the training dataset and took the average of the five model's performances. The paper didn't mention any preprocessing of the dataset, but the steps I took will be listed in a later section of this paper. The other study, conducted by Kim et al., also compared ANNs, SVMs, and SSL models on breast cancer survivability using the SEER dataset, which they trained to an accuracy of 65%, 70% and 51% for ANNs, SSL, and SVMs respectively. Listed below are the variables used by both of these studies:

stage
grade
lymph node involvement
race

age at diagnosis
martial status
primary site
tumor size
site-specific surgery
radiation
histological type
behavior code
number of positive nodes examed
number of nodes examined
number of primaries
clinical extension of tumor

A few other studies looked at the performance of machine learning methods on breast cancer survivability but with other cancer datasets. Kalafi et al. compared the performance of the ANNs and SVMs and found that ANNs performed the best with an accuracy of 88.2%; however, compared to SVMs, ANNs required more tuning of the model structure and time to complete the training. Kalafi et al. used the University Malaya Medical Center Breast Cancer Registry (UMMCBCR) dataset and used a simple neural network structure of two hidden layers with size 100 and 32 nodes along with Adam optimizer. They removed the cases that had missing values in the variables they chose, which I also followed as a preprocessing technique for my data from SEER along with their initial neural network structure as a starting point for my project.

Another study, Boeri et al. (2019), addressed the problem of predicting breast cancer survivability and compared ANNs and SVMs. They found the same results: ANNs had a better accuracy of 96.9% compared to the 95.3% of SVMs. As these studies have shown, predicting breast cancer using machine learning algorithms can lead to extremely high accuracy scores, and similar to these studies, my project is to use an ANN to predict breast cancer survivability; however, I plan to include an additional change in the model: adding an SVM in place of the last layer of the ANN.

While these studies have shown that ANNs perform well when predicting breast cancer survivability, I plan to use an ANN with an SVM because they act as a better classifier than the last layer of an ANN when classifying images. Jiang et al. (2018) examined how to combine a convolutional ANN, which is used specifically for images, and an SVM into a single model that could be trained end-to-end at the same time. When they tested this model on CIFAR-10, an image dataset of 10 different classes, the ANN and SVM outperformed a regular ANN accuracy

by 1%. The reasoning behind this is that SVMs can generalize well to new instances and can classify more complex data with many features, as their model is based on classifying data in a multidimensional hyperplane. Therefore, data with multiple features are scaled up into multiple dimensions for the SVM to find a plane, in that specific number of dimensions, that will classify the different categories (Jiang et al., 2018). The end of an ANN can also be represented as a linear transform, as the model consists of multiplication by matrices in addition to a bias offset, which can all be replaced by an SVM (R. Hartley, personal communication, Jan 28. 2021).

The model presented in Jiang et al. (2018) has been applied to detecting breast cancer in mammograms as well by classifying whether a tumor is malignant or benign (Ragab et al., 2019). Mammograms are x-ray images that are used often by radiologists to detect the presence of harmful tumors in breast cancer patients. Using an SVM classifier at the end of the ANN showed improved accuracy on the Digital Database of Screening Mammography (DDSM) dataset of up to 87.2% when compared to just a regular ANN (Ragab et al., 2019). I plan to apply this same model to the problem of predicting breast cancer survivability. Even though this study is based on classifying images, I expect an ANN and SVM combined model will also improve accuracy when using regular tabular data, or data that can be represented in a table.

Methods

Data

As mentioned before, the data I used came from the Surveillance, Epidemiology, and End Results (SEER) dataset, specifically Registry 18 which contains cases from the year 2000 to 2017. SEER has other registries available covering different years, but I chose Registry 18 because it contained the most number of cases, covering 27.8% of the US population and had a record for each 8 million tumors. In order to access the SEER dataset, a request had to be submitted on the National Institute of Health's website, and the SEER*Stat software was made available afterwards to filter and download the data.

Out of the cases available in Registry 18, I chose the cases in which the "Site recode ICD-O-3/WHO 2008" variable under "Site and Morphology" features indicated "Breast", as SEER defines those codes as based on the primary site the tumor was found. In addition, I used only cases where the patient was female to limit the variability between cases.

Below are all the variables I chose, which were selected based on the variables used in Park et al. and Kim et al. and based on which factors significantly influenced cancer prognosis.

Variable Name	**Type**
Summary Stage 2000 (1998+)	Stage
Grade	Grade

Race recode (W, B, AI, API)	race
Age recode with 1< year olds	Age at diagnosis
Primary Site - labeled	Primary site
Tumor Size Summary (2016+)	Tumor Size
CS tumor size [2004-2015]	Tumor Size
Reason no cancer-directed surgery	Site-specific surgery
Histological Type ICD-O-3	Histological type
Behavior code ICD-O-3	Behavior Code
Regional nodes positive (1988+)	Number of positive nodes
Regional nodes examined (1988+)	Number of nodes examined
Total number of in situ/malignant tumors for patient	Number of primaries
Laterality	Laterality
ER Status Recode Breast Cancer (1990+)	ER Status
PR Status Recode Breast Cancer (1990+)	PR Status
Breast - Adjusted AJCC 6th Stage (1988-2015)	Staging Variable
Breast - Adjusted AJCC 6th T (1988-2015)	Staging Variable
Breast - Adjusted AJCC 6th N (1988-2015)	Staging Variable
Breast - Adjusted AJCC 6th M (1988-2015)	Staging Variable

To determine whether each case survived the cancer or not, I used the following variables from SEER:

Survival months	Survival months
Survival months flag	Survival months flag
SEER cause -specific (or other cause of) death classification	Cause of death to cancer

The “Survival months” variable had the actual number of months the patient had survived, so I labelled any patients that had a number greater than 60 as having survived the cancer (“1” for survived, “0” for did not survive). However, the other two variables listed in the chart, “Survival months flag” and “SEER cause-specific death classification” gave more information about the death and survivability of the cases. “Survival months flag” indicated which cases had complete dates available, which would show exactly how many months the patient lived for, so I removed the cases that had “Incomplete dates are available and there cannot be zero days of follow-up” and “Incomplete dates are available and there could be zero days of follow-up” as entries for the “Survival months flag” variable.

“SEER cause-specific death classification” lists specifically what caused the death of the patient if they were dead. Some patients were dead due to causes that were not specifically breast cancer, so I also removed those cases that had “Dead (attributable to causes other than this cancer dx)” and “Dead (missing/unknown COD)” for this variable.

Preprocessing

The cases that had a missing value for any of the variables were removed from the dataset, as it would be best for all cases to be filled in completely with all values. The only variables that had missing values were “ER Status Recode Breast Cancer (1990+)” and “PR Status Recode Breast Cancer (1990+)” as these variables were missing for almost half of the given dataset but were still included because they were heavily related to breast cancer survivability. The missing values were labeled as “Unknown”, and would count as another category that is inputted into the neural network. Below is a chart with the specific values that were removed for each variable. The definitions of the values can be found at this link for Dictionary of SEER*Stat variables and this link for SEER Research Plus Data Description:

Variable Name	**Removed Value**
Summary Stage 2000 (1998+)	'Unknown/unstaged'
Grade	'Unknown'
Race recode (W, B, AI, API)	'Blank(s)', '999', '888', '996', '997', '998', '995', '994', '993', '992', '991', '990', '000'
Age recode with 1< year olds	Did not remove any values
Primary Site - labeled	Did not remove any values
Tumor Size Summary (2016+)	Did not remove any values
CS tumor size [2004-2015]	'Blank(s)', '999', '888', '996', '997', '998', '995', '994', '993', '992', '991', '990', '000'
Reason no cancer-directed surgery	'Unknown; death certificate; or autopsy only (2003+)'
Histological Type ICD-O-3	Did not remove any values
Behavior code ICD-O-3	Did not remove any values
Regional nodes positive (1988+)	98, 97, 95
Regional nodes examined (1988+)	0, 95, 96, 97, 98, 99
Total number of in situ/malignant tumors for patient	'Unknown'
Laterality	Did not remove any values
ER Status Recode Breast Cancer (1990+)	Did not remove any values

PR Status Recode Breast Cancer (1990+)	Did not remove any values
Breast - Adjusted AJCC 6th Stage (1988-2015)	'NA', 'UNK Stage', 'Blank(s)'
Breast - Adjusted AJCC 6th T (1988-2015)	'TX', 'T0', 'TX Adjusted', 'Any T, Mets', 'NA', 'Blank(s)'
Breast - Adjusted AJCC 6th N (1988-2015)	'NX Adjusted', 'Blank(s)"NA'
Breast - Adjusted AJCC 6th M (1988-2015)	'MX', 'NA'

Further variable specific preprocessing was conducted on the “Histologic Type ICD-O-3” variable, where only the first three digits of the value were used to reduce the number of categories for the variable during encoding. The TMN Staging variables, specifically “Breast - Adjusted AJCC 6th T (1988-2015)”, “Breast - Adjusted AJCC 6th M (1988-2015)”, and “Breast - Adjusted AJCC 6th N (1988-2015)” had codes that were replaced with the actual numbers each code represented.

The T staging variable describes the size and extent of the main tumor. The scale goes from 1 to 4, and the higher the number means the more it has grown; the following values, 'T0', 'Tis', 'T1mic', 'T1a', 'T1b', 'T1c', 'T2', 'T3', 'T4a', 'T4b', 'T4c', 'T4d', were replaced respectively with 0, 0, 0.1, 0.25, 0.75, 1.5, 3.0, 5.0, 6.0, 7.0, 8.0, 9.0, which gives the size of the tumor in centimeters.

The M variable describes how much the cancer has metastasized or spread throughout the body. Because all of the cases had a value of ‘M0’ which correlates to ‘no distant metastasis’, ‘M0’ was replaced with the number 0.

Lastly, the N variable gives the number of lymph nodes that have cancer. The scale goes from 1 to 3, with higher numbers meaning that there are more locations that have cancer; the following values, 'N0', 'N1', 'N2', 'N3', were replaced respectively with 0, 2, 6.5, 10, which indicate how many axillary nodes have cancer.

After the variable specific preprocessing, the data was resampled randomly so that there was an equal number of instances for patients that survived and those who did not survive. This way, the model would not be biased towards one specific class, which would inherently lead to a higher accuracy than if the data was split evenly. There were a total of 278714 alive cases and 161005 dead cases, giving a total of 439719 cases. 161005 cases were resampled from the alive cases, giving each class 161005 cases. In total, there were 332010 cases for our model to use for training and testing data.

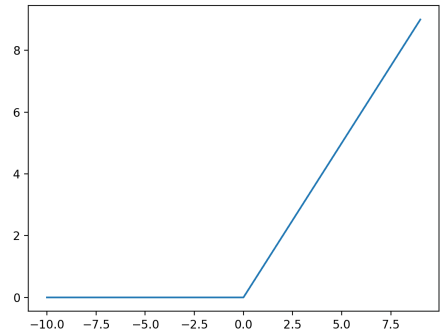
The resampled data was then split into a training set, which the model will directly train on, and a testing set, which the model will not look at at all before making predictions. This allows for an unbiased prediction and a better representation of how the model is generalizing to new cases compared to purely using training data to evaluate how well the model is performing. The split was 80% training data, 257608 cases, and 20% validation data, 64402 cases.

After splitting the data, the continuous variables in the training and testing sets were normalized separately so they had a mean of 0 and standard deviation of 1 across each variable. This prevented the testing set from influencing the mean and standard deviation used to normalize the training set. The categorical variables were also encoded, which meant that each value in the variable was now in a numerical vector representation of zeroes and ones that could be fed into the neural network. After they were encoded, the categorical variables were then normalized as well. The following were the continuous variables: 'CS tumor size (2004-2015)', 'Regional nodes positive (1988+)', 'Regional nodes examined (1988+)', 'Total number of in situ/malignant tumors for patient', 'Breast - Adjusted AJCC 6th T (1988-2015)', 'Breast - Adjusted AJCC 6th N (1988-2015)', 'Breast - Adjusted AJCC 6th M (1988-2015)', and these were the categorical variables: 'Summary stage 2000 (1998+)', 'Grade', 'Race recode (W, B, AI, API)', 'Age recode with <1 year olds', 'Primary Site - labeled', 'Reason no cancer-directed surgery', 'Histologic Type ICD-O-3', 'Behavior code ICD-O-3', 'Laterality', 'ER Status Recode Breast Cancer (1990+)', 'PR Status Recode Breast Cancer (1990+)', 'Breast - Adjusted AJCC 6th Stage (1988-2015)'.

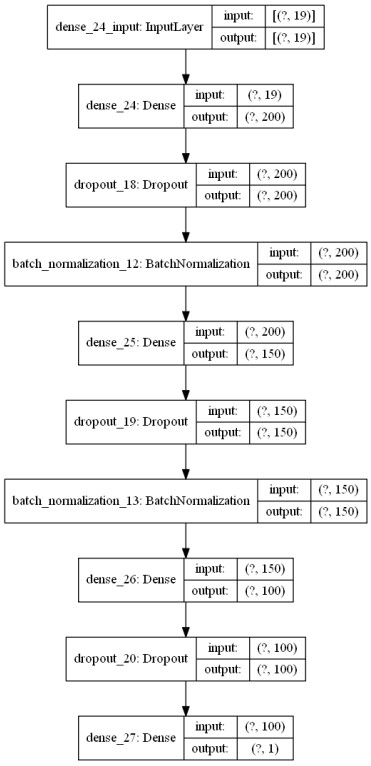
Regular Neural Network Structure

After finishing preprocessing, three different models were trained: a regular feed-forward neural, a feed-forward neural network with a linear SVM in the last layer, and a feed-forward neural network with a RBF SVM in the last layer.

The regular feed-forward neural network consisted of 3 hidden dense layers with 200, 150, and 100 nodes respectively. Several trials were run with different numbers of nodes for each layer, and this configuration had the highest accuracies. The last layer consisted of one node that gave the prediction for which of the two classes the patient would be in (whether they were alive or dead). The rectified linear activation function, or ReLU, was used as the activation function after each dense layer, which changes all of the negative output values from the neural network to 0. Dropout layers with a 30% dropout rate and batch normalization layers were inserted after dense layers as well to help with regularization and to prevent overfitting, so the model doesn't purely memorize the training data. Dropout layers allow the neural network to randomly choose a percentage of the nodes to not include in the training for each batch of cases, and the batch normalization layers scale the output of the neural network to a mean of 0 and standard deviation of 1. In addition, binary cross entropy loss was used, as it's the most common loss function used for binary classification problems.



Img. 1. From https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks. Graph of the rectified linear unit function. The output of the graph for values less than 0 is 0.



Img. 2. Structure of regular feedforward network with three dense layers, dropout, and batch normalization layers.

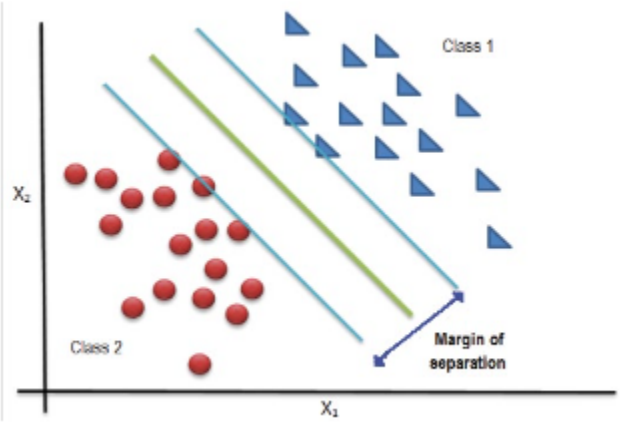
Linear SVM and Neural Network Structure

In order to implement the linear SVM in the last layer, the loss function was changed to hinge loss during training and a kernel L2 regularizer was added to the last layer (Yichuan, 2013).

The way a linear SVM functions is it tries to find the largest margin to linearly separate the given labels. As shown in Image 3, the line is used to separate the red circles and blue triangles into two classes and creates a margin of separation as wide as possible so the model can easily differentiate the two classes. Equation 1 gives the constrained optimization problem the SVM is trying to solve and indicates how right or wrong the model is based on its predictions. Through training, the model is trying to minimize this value, which would maximize the accuracy of its predictions. Minimizing this value would also allow the SVM to find a better separator with a theoretically larger margin.

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{n} \ell(y_i, w' x_i)$$

Eq. 1. Images from https://cvstuff.wordpress.com/2014/11/29/latex-l_1-versus-latex-l_2-loss-a-svm-example/. Constrained optimization problem for the SVM to minimize in order to minimize the loss of the model.



Img. 3. Image from Kalafi et al. Graph of two-dimensional linear classification problem using an SVM.

The first term in Equation 1 serves to maximize the distance between the two classes, and the second term is the loss function whenever the SVM misclassifies a point or when there is a datapoint inside the margin of separation. The loss function is calculated given the ground truth, target label *y*, and the predicted label given by the weights of the neural network, *w*, transposed and matrix multiplied by the input features, *x*.

The name for this loss function used with SVMs is hinge loss. As shown in Equation 2, hinge loss takes the maximum value between 0 and 1 minus the predicted multiplied by the target labels.

$$\ell(y_i, w' x_i) = \max[0, 1 - y_i w' x_i]$$

Eq. 2. Image from https://cvstuff.wordpress.com/2014/11/29/latex-l_1-versus-latex-l_2-loss-a-svm-example/. Equation for hinge loss.

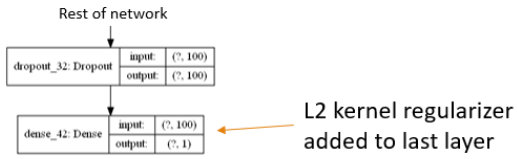
The two terms in Equation 1 added together can be interpreted as the total loss of the SVM to minimize and this combination of values can be replicated in the neural network, which is the same as adding an SVM to the last layer of the neural network.

Equation 3 shows how the total neural network loss can be similarly represented using the two terms in the total loss for the SVM. The loss function was changed to hinge loss function -- Keras has an implementation of hinge loss -- from binary cross entropy loss and, to include the second term of Equation 3, an L2 regularizer of value 0.01 was added to the last fully connected layer as well to complete the SVM.

$$\text{Loss}(w, x) = \text{DataLoss}(w, x) + \frac{1}{2} c \|w\|^2$$

Eq. 3. Image from https://blog.janestreet.com/l2-regularization-and-batch-norm/. Equation for the total loss for neural network.

L2 regularizers add the sum of the squared weights to the loss function, as shown in Equation 3 with the addition of the absolute value of squared weights. The effect of the L2 term is that the weights of the neural network decay proportionally towards zero and helps to reduce overfitting. This is similar to how in an SVM, the goal is to maximize the margin of the linear separator so the model will generalize well to new data points.



Img. 4. Structure of neural network with linear SVM in the last layer.

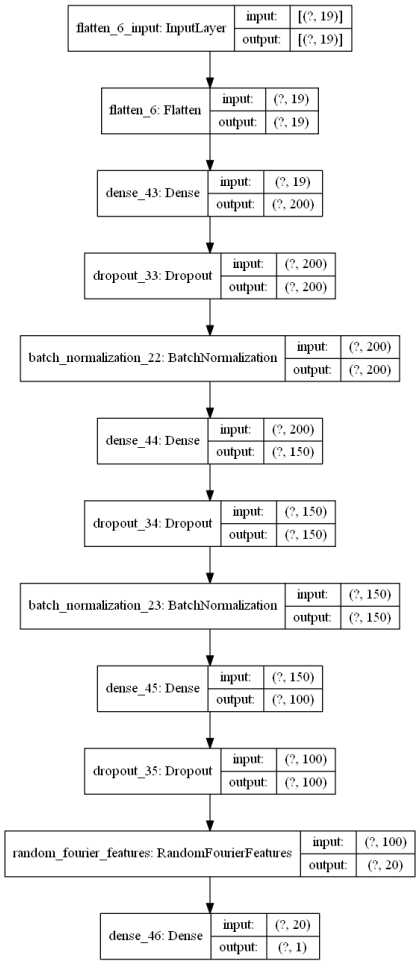
RBF SVM and Neural Network Structure

An RBF SVM is very similar to the linear SVM in terms of how it works conceptually, but it can model more complex data forms and often has a higher accuracy compared to linear SVMs.

The RBF SVM in the last layer of the neural network was implemented by adding a layer that projects the inputs into a higher dimensional space. As complex data is plotted onto a higher dimensional space, their separator to classify the data becomes less complex or more “linear” in structure which then makes it easier to differentiate the classes. The number of dimensions for the layer’s output was set to 20 using Jiang et al. as a reference, and the parameters of the layer were randomly resampled from the Gaussian distribution. Further trials with a higher and

lower number of dimensions of the layer's output did not lead to higher accuracies, so the final number for the output was set to 20.

Tensorflow includes an implementation of this layer called RandomFourierFeatures which was placed right before the layer with the one node at the end of the network that would predict whether the patient was alive or dead. This layer is used to "kernelize" linear models by applying a non-linear transformation to the input features and then training a linear model on top of the transformed features. The linear model is the layer with one node. When used with the hinge loss function with Keras, the composition of the RBF SVM was complete.



Img. 5. Structure of neural network with RBF SVM using RandomFourierFeatures.

Training

With the regular feed-forward neural network, various models were trained using different combinations of learning rates, optimizers, and dropout rates. After finding the highest performing regular neural network structure and configuration, the same set of specifications were applied on the model with a linear SVM and RBF SVM to see if there would be an improvement in accuracy.

One cycle learning rate appeared to work the best with a maximum value of 0.001 when compared to a standard learning rate and exponential learning rate, and Adam, or adaptive moment estimation optimizer, performed the best compared to regular standard gradient descent, known as SGD, and root-mean-squared propagation, known as RMS prop.

To determine the best values for hyperparameters, a grid search was conducted to find dropout rate. Searching through a space of 0.0 to 0.9 dropout rate with 0.1 intervals showed that a 0.3 dropout rate had the highest accuracies. The grid search used 3-fold cross validation as well, where there were three models created and trained on three separate sections of the training data. The average of the accuracies for the three models for a specific dropout rate then gave a better representation of the model structure's accuracy.

He initialization was used when creating the initial weights for the model to allow the gradients to remain stable during training by making the weights for each layer have a variance that depended on the number of nodes for that layer.

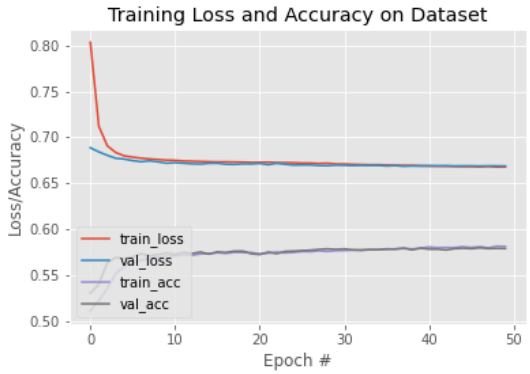
During training, the following callbacks were used in Keras: ModelCheckpoint, OneCycleScheduler, and Tensorboard. Callbacks are methods that can be called during training to provide information about how the model is performing. ModelCheckpoint is a callback provided by Keras, and it saves a version of the model in the middle of training depending on what value the checkpoint is based upon. For these models, the callback was based upon validation loss, so the model with the lowest validation loss was saved across each training session. OneCycleScheduler was an external callback outside of Keras based on an implementation from the Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow textbook by Geron (Chapter 11), and it allows the learning rate to change based on where the model is during training. Finally, Tensorboard is also a callback provided by Keras, and it graphs the accuracies and losses of the model in an interactive application that can be accessed on the local computer. Tensorboard was useful to visualize how well the models were performing, but it was difficult to find a way to implement this with auto cross validation functions in scikit-learn without having to manually implement cross validation. Graphs created using the matplotlib library were also used for visualization.

The final models were trained with 50 epochs, which means the model saw the training data 50 times total, and a batch size of 128. All three models showed a relatively flat plateau at the end of 50 epochs, so the training ended there.

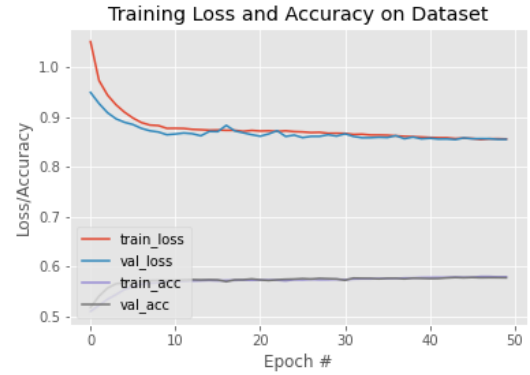
Results

Using a neural network with these parameters and 3-fold cross validation, the model was able to reach an average accuracy of 57.85%, 57.73% when combined with a linear SVM, and 57.39% when combined with the RBF SVM. The graphs below plot the training and validation accuracy and loss of the best run for the three models.

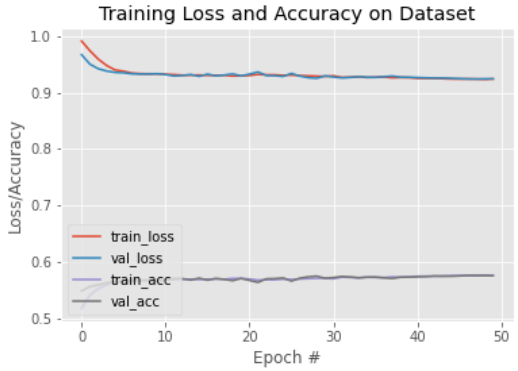
Model Type	Model 1 Accuracy	Model 2 Accuracy	Model 3 Accuracy	Average and Standard Deviation
Regular NN	0.5784	0.5790	0.5781	57.85% (+/- 0.04%)
Linear SVM	0.5777	0.5770	0.5773	57.73% (+/- 0.03%)
RBF SVM	0.5739	0.5754	0.5725	57.39% (+/- 0.12%)



Graph 1. Regular Neural Network.



Graph 2. Linear SVM with Neural Network.



Graph 3. RBF SVM with Neural Network.

Due to time constraints, the accuracy of the neural network was not as high as originally intended; however, the highest accuracy for previous studies that used the same SEER dataset and a neural network reached an accuracy of 65% (Park et al., 2013, Kim et al., 2013). Although the model wasn't able to reach a higher accuracy, it was still trained to a standard relatively close to that of previous projects.

Other model configurations trained to a training accuracy of 60%, as shown in the graph below, except the validation accuracies eventually worsened. These models overtrained, and adding dropout layers and other regularization techniques helped prevent this from happening.

Graph 4. Regular Neural Network with one dropout layer.

This specific model only had dropout layers before the last layer with one node when the other models had dropout layers before every dense layer.

Looking at the results of the neural network and SVM combined models, they did not improve the accuracy of the neural network as originally hypothesized. This could be due to how the data used was not complex enough, as some data was missing from the SEER dataset that was used in other papers, such as marital status and whether the patient was exposed to radiation

(Park et al., 2013). In addition, the lower accuracy of the normal neural network could have influenced or worsened the performance of the model when it was combined with a linear and RBF SVM.

Another possibility for the decreased accuracies could simply be that the SVM is redundant or not needed with the neural network specifically for this problem. Previous papers used CNNs with SVMs, and this project used a normal neural network; the non linear transformation the SVM gives is already accounted for in the neural network activation functions, such as ReLU, so this additional layer could have messed with the output of the network when it wasn't needed.

Despite the lower accuracies, neural network and SVM combined structure can still be useful on this problem given more work on improving the accuracy of the normal neural network and trying more hyperparameter searches regarding the number of layers, number of nodes, and the learning rate of the normal neural network.

Additionally, one result from this project was that the combined linear SVM model had a smaller standard deviation, which could indicate that the models are more consistent, though the difference in standard deviation between the models is very small.

Extensions

Including the steps previously mentioned to improve neural network accuracy, there are also possible extensions and other areas to investigate for this project.

Looking forward, it would be interesting to look into the correlation between different variables to see which variables influence breast cancer survivability the most and therefore which information healthcare physicians should pay most attention to.

With machine learning models, the input variables to the algorithm play a major component in determining how well the algorithm will perform. Ganggayah et al. (2019) focused on finding which characteristics of breast cancer were the most influential in predicting the survival rate using random forests, a machine learning algorithm consisting of several "trees" that make predictions using yes or no questions. Through visualizing the structure of the random forest model, Ganggayah et al. (2019) found that stage classification, tumor size, positive lymph nodes (how many lymph nodes were affected by cancer), treatment type, and method of diagnosis variables carried the most weight in determining survivability. Kalafi et al. (2019) also determined the importance of variables using random forests and looking at each variables' relationship to survivability. They found tumor size, cancer stage, age, and positive lymph nodes to be the most important factors. These are variables that were verified could be used when training the models for this project compared to other factors available in the SEER cancer database. Depending on the results from the three models and the data used, looking at which variables influenced the normal neural network compared to the SVM included networks would demonstrate which factors are the most important and which are more subjective.

Another model structure to explore is having the neural network and SVM trained separately, taking the features from a layer in the neural network, and feeding those features into the SVM to further classify the labels. The neural network would primarily serve as a model to generate larger dimensional features for the SVM to externally train on, and it would be interesting to compare the difference in accuracies for this model structure and the three used in this project. The first steps for this structure were coded but weren't finished due to time constraints.

In addition, while looking at the activations from the output of the normal neural network, most of the activations had an output of 0. This meant that the model didn't have much information to determine whether the patient would be dead or alive during training, so it would be great to find a way to prevent the output from converging to 0.

Lastly, it would be worthwhile to experiment with resampling methods such as SMOTE, or Synthetic Minority Oversampling Technique, which synthesizes new examples from existing examples. This way, all the data can be used versus needing to resample classes to have the same number of cases and leaving some cases out like this project. imblearn is a good Python library that could be used for this type of data processing.

Implementation

This project was coded in Python 3.8.0 using an Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz. The Computer Systems Lab-provided JupyterHub was also used with the Default job profile of 4 CPUs and 8 GB RAM to train models.

Preprocessing was conducted using the NumPy (version 1.19.2) and pandas (version 1.1.3) libraries. Tensorflow (version 2.3.0) with Keras (version 2.4.0) was used as the deep learning framework and scikit-learn (version 0.32.2) was used for cross validation and matplotlib (version 3.3.2) for visualization.

The preprocessing Jupyter Notebook lists all the steps used to prepare the data. Run the cells in the order shown on the notebook, and the output should be two files for the training and testing sets.

The training Jupyter Notebook contains the structure for training the three network structures with 3-fold cross validation. Run the cells in the order shown in the notebook, and the neural network should be successfully trained then. Double check that the paths work for your computer and that you have the data available for the program to read it.

More information about the data used and definitions for the variables can be found here:

- SEER Registries and Database Information
- Requesting Access to SEER Data & Downloading SEER*Stat software
- SEER Research Plus Data Description (includes all the variables and explanations for values used for variables)

- Dictionary of SEER*Stat Variables (comprehensive guide for all variables in SEER database)

Lessons Learned

Training the models was difficult due to the wide range of parameters and hyperparameters that could be changed. For future projects, make sure to write down everything that you've done and where you've left off, as this will make it so much easier to understand your thought process if you end up taking a break in the middle of your project.

When training models, try to use cross-validation or train the model multiple times using the same configuration to get a more generalized representation of the model, as the randomness in initializing the weights could influence the model accuracy.

Citations

Boeri, C., Chiappa, C., Galli, F., De Berardinis, V., Bardelli, L., Carcano, G., & Rovera, F. (2020). Machine Learning techniques in breast cancer prognosis prediction: A primary evaluation. *Cancer medicine*, *9*(9), 3234–3243. https://doi.org/10.1002/cam4.2811

Geron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly.

Kalafi, E. Y., Nor, N., Taib, N. A., Ganggayah, M. D., Town, C., & Dhillon, S. K. (2019). Machine Learning and Deep Learning Approaches in Breast Cancer Survival Prediction Using Clinical Data. *Folia biologica*, 65(5-6), 212–220.

Kim, J., & Shin, H. (2013). Breast cancer survivability prediction using labeled, unlabeled, and pseudo-labeled patient data. Journal of the American Medical Informatics Association : JAMIA, 20(4), 613–618. https://doi.org/10.1136/amiajnl-2012-001570

Park, K., Ali, A., Kim, D., An. Y., Kim, M., Shin, H. (2013). Robust predictive model for evaluating breast cancer survivability. Engineering Applications of Artificial Intelligence, 26(9), 2194-2205, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2013.06.013.

Ragab, D. A., Sharkas, M., Marshall, S., & Ren, J. (2019). Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*, *7*, e6201. https://doi.org/10.7717/peerj.6201

Tang, Y. (2013). Deep Learning using Linear Support Vector Machines. arXiv: Learning.Kernel

Jiang, S., Hartley, R., & Fernando, B. (2018). Kernel Support Vector Machines and Convolutional Neural Networks. 2018 Digital Image Computing: Techniques and Applications (DICTA), 1-7.

https://blog.janestreet.com/l2-regularization-and-batch-norm/

https://cvstuff.wordpress.com/2014/11/29/latex-l_1-versus-latex-l_2-loss-a-svm-example/

https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Acknowledgements

Thank you to Dr. White for providing technical advice when about neural network training and for helping me formulate my project idea. Thank you to TJHSST's Computer Systems Lab for providing computing resources for me to train my models on, and my classmates for giving valuable feedback on my previous class presentations.