Predicting Breast Cancer from Tumor Cell Nuclei

Seth Singson-Robbins
Fordham University
ssingsonrobbins@fordham.edu

Abstract – This paper investigates the design and analysis of a classification model that determines, based on extracted cell nuclei from patients with breast cancer tumors, if the tumor is benign or malignant (cancerous). Classification is based on the nuclei's size and other physical attributes with several metrics per attribute. Model includes utilizing Principal Component Analysis (PCA) to help with feature selection along with determining weights and patients used to influence the boundary line through Search Vector Machine learning (SVM). Parameters include the PCA Kernel, number of PCA components, SVM Kernel, and SVM slack variable leading to 240 different models that are tested through validation data. The final model was tested on final test data and had an accuracy rate of 91%.

I. Intro

According to BreastCancer.org, about one in eight women will develop invasive breast cancer in the United States. Even though breast cancer has become much more treatable over the years, it is still estimated that about 43 thousand women will die from the disease with the success of beating the cancer being significant by knowing earlier to prevent the cancer from spreading. [(1)] However, the surgery is very invasive and, along with the financial constraint on the patient, insurance company, and medical system at large, can have significant complications including bleeding, infection, fatigue, and even the need for chemotherapy after the removal of the tumor even if not cancerous. [(2)] This is a cause for concern due to 60-80% of tumors are benign [(3)], or not cancerous, and sometimes removal not even being necessary, so less invasive methods to confirm outside the surgical removal of these tumors would be preferred.

Being able to look at a small number of cells to test for cancer could prove to be a much less invasive method to determine if a lump is cancerous and needs to be removed. If easy to extract and reliable as an initial source, it could be a great first step in the examination process while decreasing the number of unnecessary surgeries, helping alleviate resources for the medical industry, reducing costs for patients and insurance companies, and preventing the patient from having to suffer from the complications of surgery and instead extracting a few cells from the tumor.

II. Methodology

This section looks at data and its transformations to the data before being inputted into the classification model. This includes information on data collection, pre-processing transformations on the data, and how the test and training data was set up to help with validation.

A. Data Collection

Data was collected from the Kaggle database containing 569 examples of patients who had a tumor and were able to identify whether the tumor was benign or malignant. [(4)] Along with diagnosing if a tumor is malignant or not, the data contains 30 features, 10 attributes of attributes based on the tumor cells' nucleus including the size and texture, with three performance metrics of each characteristic (the average, standard error, and 'worst' (ex. for size is the mean of three

largest cells)) of each characteristic. Within the dataset, 212 or 37% of the patients have been diagnosed as having a malignant tumor, which means this is an imbalanced dataset, albeit not a significant one. Discussions on how the data was rebalanced will be addressed in the test and training section in part C.

B. Preprocessing

Given the nature of the data and the requirements of data when going into classification models, some transformations of the dataset in the dataset for the models to run effectively and efficiently while having stronger predictability. The SVM model requires the classes to be divided as values of -1 and 1, so the predictor feature that determines the diagnosis was updated so that examples with the value 'M', for malignant', was changed to 1 while the examples with the value 'B', for benign, was updated to -1. This is logical since malignant cases are the class the model is trying to identify.

Additionally, the data has varying ranges based on the feature which could lead to bias due to the differences as well as longer run times for the model to find the optimal weights. To account for this, the data was normalized so that each feature has a mean of 0 and a variance of 1 among the examples within the full dataset.

C. Training and Evaluation

With several hyperparameters being tested on the model, which will be discussed in parts III and IV, the total variations of these hyperparameters required a multitude of different models all looking to see which is the best model to accurately determine if a tumor is malignant from its cell nucleus. To help evaluate the numerous models, 20% of the dataset was separated from the training data to help with the final evaluation. This was done through the SKLearn package with the function train_test_split. [5] Of the 20% that was taken out of the training data, 50% (10% of the total dataset) was used for validation of the models while the other 50% (10% of the total dataset) was used for the final performance analysis of the model chosen to be the final predictor.

Additionally, due to the data being imbalanced with more benign examples than malignant examples, random oversampling of the malignant examples on the training data was implemented to have the malignant and benign tumors have the same number of training examples. This was done through imlearn's RandomOversampling function.[6]

III. Feature Selection and Principal Component Analysis

As explained in the data collection section, the data makes up 30 features that describe features of the cell nucleus with 10 features with 3 different performance metrics. Given that there are several columns for the same feature and how there are large similarities between the features, there is a significant amount of co-variance which could lead to a loss of accuracy due to overfitting from the curse of dimensionality. This is especially a cause for concern due to the small number of data points in the dataset itself which, in combination with the large number of features, exponentially increases the chances of overfitting and a loss of accuracy when implemented on new examples. Feature selection is a way to reduce the number of features the data utilizes for modeling while minimizing the differences between the datapoints that helps the model distinguish between a malignant and a benign tumor. Feature selection was implemented through Principal Component Analysis (PCA) which will be explained in detail below including additional information on the two hyperparameters varied and tested within the models: PCA Kernels and the number of features utilized after the dataset has been transformed.

A. What is PCA?

Principle Component Analysis is a type of factor analysis that utilizes the current features and applies linear transformation of the dataset based on the covariances between the data's features reconstructing new data points. The idea is that it maximizes what it can with as few features as possible and ranks the new features by the total explained variance within the model.[7] The entire table can be transformed by a small number of components transforming each of the data points with the same components with all the components being orthogonal to each other.[8]

B. PCA Kernel

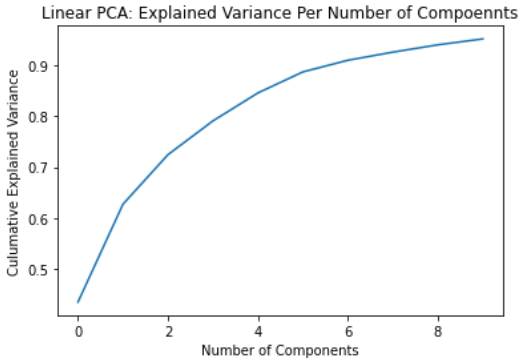
For the typical PCA transformation, the data utilizes only linear data, or a linear transformation of each of the features to transform the table. However, there are other options that look at the data at a higher number of dimensions through kernels. This helps separate features that cannot be perfectly separated from the typical linear PCA alone. One risk from implementing kernels into PCA is, due to the increase in the number of features created from the kernel, the risk of overfitting the data and hurting accuracy increases when implemented into machine learning models which is why these will all be tested. Along with the typical linear kernel, the following three kernels were used through the SKLearn's KernelPCA[9] function with the formulas included (x and y are two features being compared to each other which will be done for every combination of two features):

Kernal Functions:[10]

- Poly Kernel: $K(x, y) = (x \cdot y)^3$
- Radial Basis Function Kernel (RBF): $K(x, y) = \exp exp\left(-\frac{|x-y|^2}{2 \cdot \sigma^2} \right)$
- Sigmoid Kernels: K(x, y) = tanh $(x \cdot y)$

C. PCA Components

When transforming the data, the number of components is usually the same number of features as the original dataset for linear PCA and even more features when utilizing other kernel PCAs. The difference between the transformed data is that the features are optimized and ordered by the amount of variance captured between the examples with the training set. The chart below shows the amount of captured variance cumulated by the top n number of features for the linear kernel. Note that this was taken from the SKLearn's PCA function,[11] which is the equivalent of the linear kernel in KernelPCA, and the other kernels not having the captured variance due to limitations related to the higher dimensionality.



The first component makes up 45% of all the variance between the dataset while the top 10 components make up 95% of the dataset. This means that for 10 components chosen it is losing 5% of the information the dataset explains about its examples. Choosing the number of components is something that needs to be taken into consideration and is a hyperparameter for this model. A smaller number of components will lower the chance of overfitting but will mean some of the knowledge from the data will not make it into the machine learning model. However, having too many features kept into the model could result in overfitting, defeating the purpose of utilizing PCA. For every hyperparameter combination used for testing, the number of components will be included in the testing from the top component through the top 10 components.

IV. Support Vector Machine Model

For the actual method of predicting whether a tumor is benign or malignant, the model utilized is Support Vector Machines. A lot of the cell nuclei were hard to determine just from the features which means that the ones that are near the decision boundary are the most important. Support Vector Machine excels at this because, unlike logistic regression and most other classification models, this model looks at only the examples that are close to the possible boundary line to shape where the boundary line will divide up the classification of future data points instead of other examples that clearly are in one class over the other.[12] Support Vector Machines (SVM) were made utilizing the NumPy[13] package and will have further discussions into the formation of the model including the use of Dual Kernel learning, how alphas are determined, and the two hyperparameters that are tested in the models: SVM Kernels and the slack variable used to optimize the model decision boundary.

A. Dual Kernel Model

Support Vector Machines have several ways of optimizing towards finding the weights and support vectors utilized for classification modeling. The method used for this analysis was through the Dual Kernel Model. Sometimes SVM compares the dataset through a primary LaGrangian multiplier, focusing on the features to determine the optimal weights. Each data point is then optimized as a separate component of the function and added together in the minimization technique. Dual Kernel looks at the relationship of the two data points with a multiplicative formula that utilizes the variance of each data point couple to optimize the weights of each alpha or support vector instead of focusing on the weights themselves. [12] The final weights are based on the values of the weights of the support vectors and the values of both their

features and classifications for the final model weights. This allows us to look at higher dimensionality of the data through Kernels than just the typical linear approach used. The following LaGrange multiplier is utilized to determine the appropriate alpha values.

$$\sum_j a_j \; - \; \frac{1}{2} \sum_i \sum_j y_i \, y_j \, a_i \, a_j \, K(x_i, \, x_j)$$

The model mainly looks at finding the maximum LaGrange multiplier value based on the alpha values of each data point. The formula is interesting because it compares every single data point to each other, a total of n^2 comparisons, and utilizes both its classification value and kernels as part of its analysis.

To find the maximum, the model will search for the optimal alpha values through the gradient descent technique meaning that it will learn the discrepancy between the two data points for every combination and round and run through the data through gradient descent until it finds the optimal final alpha values for the formula. The follow shows the gradient descent formula and the calculations showing how the LaGrange multiplier was derived to be implemented into the gradient descent:

$$a_i = \; a_i \; + \; \frac{d}{d\alpha} L(a, \, x, y)$$

$$\frac{d}{d\alpha} L(a, \, x, y) \; = \; \frac{d}{d\alpha} \left(\sum_i a_j \; - \; \frac{1}{2} \sum_i \sum_j y_i \, y_j \, a_i \, a_j \, K(x_i, \, x_j) \right)$$

$$= \; \sum_i \frac{d}{d\alpha} a_j \; - \; \frac{1}{2} \sum_i \sum_j y_i \, y_j \; K(x_i, \, x_j) \; \frac{d}{d\alpha} a_i \, a_j$$

$$= \; \sum_i \frac{d}{d\alpha} a_j \; - \; \frac{1}{2} \sum_i \sum_j y_i \, y_j \; K(x_i, \, x_j) \; \frac{d}{d\alpha} a_i \, a_j$$

$$= \; 1 \; - \; \sum_i y_i \, y_j \; K(x_i, \, x_j) \, a_j$$

Within the model, NumPy[12] was used to expedite these calculations with the use of NumPy arrays. As proven in linear algebra, multiplying the transposition of the array by the same array shows the dot product combination of every combination of the items in the array. So $Y = y_i \bullet y_j$ and K is an array showing the kernel (explained further in part C) for each point within $K(x_i, x_j)$.

Some other things to take into consideration include the learning rate and the number of iterations of the model. 0.001 was used since it got to the optimal LaGrange multiplier value while not exponentially increasing the alpha values optimizing the use of the final model. The model runs 100 times as through testing it looks to converge successfully for all the models without a significant number of unnecessary iterations.

The final model utilizes the normal linear boundary $y = w^T \bullet x$ with the SVM calculating the weights with the formula $w = \sum_i a_i \cdot x_i \cdot y_i$ with i being each data point in the training data. When new examples are inputted for classification within the model, the new examples are transformed with the PCA model and utilize the final weights that were optimized from the

model to get a number signifying its distance from the linear boundary. If the score is equal to or above 1, then the examples will be labeled as ‘malignant’. If it is equal to or below -1, then the examples will be labeled as ‘benign’. If the score is between -1 and 1, the example will be labeled as ‘unidentified’. Due to the nature of SVM, there should be little to no ‘unidentified’ examples.

B. Alpha Regularization

One of the boundaries of the dual kernel model is that the following function must always hold true for all examples[12]:

$$\sum_i a_i \cdot y_i = 0$$

To ensure this happens, the function regularize_alphas was created to ensure that this rule is true at the end of every iteration of gradient descen. Since the alpha values are all 0 or positive, the function sums up all the alpha values that come from patients with benign tumors (with yi values of -1) and separately adds up all the alpha values of patients with malignant tumors (with yi values of 1) then looking at the ratio between the two. It then applies this ratio to every alpha from patients with malignant tumors through multiplcataion ensuring that its new summation of the alphas from the patients with malignant tumors is equal to the alphas from patients with benign tumors. The code for this is below.

```
def regularize_alphas(y_train, a):
    # pushes down examples with negative a's to 0
    a = np.maximum(a, 0)

    if np.abs(np.minimum(a * y_train,0)).sum() > 0:
        sum_ratio = np.maximum(a * y_train,0).sum() / np.abs(np.minimum(a * y_train,0)).sum()

        # multiplies the ratio with the summation of negative class so equal to summation of positive class
        a = np.multiply(a, np.where(y_train == -1, sum_ratio,1))
    else:
        a = 0 * a #set all support vectors to 0 since model is void (all examples in 1 class)
    return a
```

One note in the code is that, if all the patients in the training data have an alpha value of 0 (thus not support vectors), then that means all the support vectors are in the negative class. This is not logical for the model since we are trying to differential patients between the two classes as the point of classification. This function will also throw an error if this occurs, which is optimal, causing the gradient descent to error out and not providing a model to run future examples on. The code that creates the models will set all the alpha values to 0 causing the weights to also be 0 which helps the models be easily identifiable as not converge and thus can easily be removed from consideration.

C. SVM Kernel

Just like the PCA Kernels, SVM is not always optimal when only linearly separating the examples and finding the optimal support vectors. To find more dynamic cuts of the data, kernels can be implemented past the typical linear separation to look at the data at a higher dimensionality. This model tests the radial basis function (RBF) like PCA to test if higher

dimensionality will be a more accurate representation of the dataset. [12] While it could be more accurate, if a linear method is the optimal model the other Kernel could risk overfitting due to extra features which is why testing both options is the optimal solution. Below are the formulas utilized for the model (Not that σ^2 is a hyperparameter that was set to 1 since the features all were normalized to have a variance of 1).

- Linear Kernel: $K(x_1, x_2) = (x_1^T \bullet x_2)$
- RBF Kernel: $K\left(x_1, x_2\right) = -\frac{1}{2\sigma^2}\left(\left|x_1 - x_2\right|^2\right)$

D. SVM Slack Variable

When the SVM model creates the classification boundary, one requirement is that the function $y = w^T \bullet x$ must be greater than 1 and less than -1 for every patient in the training set so that it can be confidently classified at every point. When support vectors are close to this boundary, it could push the final classification boundary in strange and suboptimal directions hurting accuracy for future patients. Slack variables can be introduced to the model to help loosen this restriction. Slack variables look at the error rate of incorrectly labeled examples, or patients in this model, and allow it to be correctly labeled if it is near the boundary. [12] For example, if the function comes out to 0.8, we might allow the tumor to be labeled as 'malignant' if the slack variable is at 0.2.

When implementing the slack variable, a penalty multiplier is given to allow a maximum amount of influence that the slack variable can give to the model. The higher the C value, the more influence that the slack variable can give. Given this is determining if people will have surgery to extract their lump, we do not want it to have a high enough influence that it will cause significant errors in accuracy. This model is given 0.2 since it will allow up to 20% of the error to be excused from classification.

Since we are doing gradient descent to ensure that the model accurately labels every patient in the training data, this change in classification will have significant changes to the final model when slack variables are used. We will utilize both the linear slack variable and the squared slack variable. The below shows the final classification model and the learning rate that is added at the end of the gradient descent function (ξ is the error rate):

- Linear: $y_i \bullet (w^T \bullet x_i) \geq 1 - 0.2 * \xi$
  - Learning:

$$\frac{d}{d\alpha} C * \xi$$
$$= C * \frac{d}{d\alpha} (1 - w^T \bullet x)$$
$$= 0.2 * \frac{d}{d\alpha} (1 - (\alpha_i \bullet x_i \bullet y_i)^T \bullet x)$$
$$= 0.2 * - (x_i \bullet y_i)^T \bullet x)$$

- Squared: $y_i \bullet (w^T \bullet x_i) \geq 1 - 0.2 * \xi^2$
  - Learning:

$$= \frac{d}{d\alpha} C * \xi^2$$

$$= C * \frac{d}{d\alpha}(1 - w^T \cdot x)^2$$
$$= 0.2 * \frac{d}{d\alpha}(1 - (\alpha_i . x_i . y_i)^T \bullet x)^2$$
$$= 0.2 * (-2 * (1 - (\alpha_i . x_i . y_i)^T \cdot x) * (x_i \bullet y_i)^T \bullet x))$$
$$= - 0.4 * ((1 - (\alpha_i . x_i . y_i)^T \cdot x) * (x_i . y_i)^T \bullet x))$$

V. Results

As previously written, both PCA and SVM will be utilized for the transformation and modeling of this dataset to determine, based on a patient's cell nuclei from the breast tumor, whether their breast tumor is benign or malignant. The following hyperparameters will be implemented for testing and validation:

- PCA Kernel: Linear, Poly, RBF, Sigmoid
- PCA Components: 1 – 10 used
- SVM Kernel: 'Linear', 'RBF'
- SVM Slack Variable: 'None', 'Linear', 'Squared'

From a combination of all the hyperparameters above, every combination leads to a total of 240 models that will be all be tested on the validation data with the model and hyperparameters with the best accuracy utilized for the final round of analysis and accuracy score through the test data. The models were saved with results and key attributes in the predictionModel object.The results will include a look at what models did not successfully run, the run time of the models, the number of support vectors used on different models, how the accuracy performed for each model, and which model was chosen to be the model used for the final analysis.

A. Model Success

For a SVM model to run successfully, there needs to be a mix of positive and negative classes that have support vectors to determine the decision boundary. When all examples from either class all have alphas of 0, then the model will not be successful at identifying new examples since both classes are not a part of the decision. To account for this, the regularize_alphas function sets all alphas to 0 if this occurs so that the model will automatically have an accuracy of 0% and can be identified as a model that did not set a successful boundary. Of the 240 models that ran, only 1 model (Hyperparameters: 'Poly' PCA kernel, 2 PCA Components, 'Linear' SVM kernel, 'Squared' slack variable) was not successfully able to run.

B. Run Time

The models need to run efficiently since 240 models are tested. Along with the SVM efficiencies utilizing arrays instead of for loops to match the dual kernel model and gradient descent, the models are iterated by running and saving the PCA model and transformation before running every iteration of the SVM Kernel and slack variables. This saved time from not having the run the PCA model 6 times for every iteration of the PCA. The predictionModel object also saves the components of the PCA model to transform future examples that are pushed into the model for classification including the validation and test data. The total running time of the

models is 7 minutes and 39.3 seconds which means each model on average takes 1.9 seconds to run.

C. Support Vectors

The SVM model's gradient descent learns which support vectors to utilize and how much influence each one gets in the final weights of the model through the alpha value. Appendix C and D show the number of support vectors that were used to determine the decision boundary of each model. The range of the number of support vectors used is large, from 2 support vectors (which means 1 example from each class was used) to 569 support vectors (all the examples in the training data). One interesting insight is that the correlation between the number of support vectors and the accuracy is at 0.8 which means that there is a strong relationship between more support vectors and accuracy for each of the models although as shown in the next section does not mean the model with the most support vectors is the best one to use for final modeling.

D. Accuracy

The model's main key performance indicator looks at the accuracy of the model by running validation data, or data that was not in the training set and set separately from the test data, to determine the best model for each combination of hyperparameters for each model. For a patient to be correct classified, the absolute value of the score from the model needs to be above 1 (or it will be considered 'unidentifiable') and need to be the same sign (positive or negative) of its classification (or it will be labeled 'incorrect'). A few interesting insights came from the results of the accuracy from the models. The first insight is that no examples were identified as unidentifiable in any of the models. This makes sense since the model works to minimize this from happening. The second insight is that in general the slack variables tend to worsen performance of the models which means that giving some slack from incorrectly labeled examples in the training data in general is leading to overfitting. The last insight is that the polynomial PCA kernel had the best performing models showing that transforming the data through a lens with more dimensions is pertinent to correctly analyze the model.

Appendix A and B show the accuracy of all the models from the validation dataset. The accuracy of the models was on average at 51.97% with a range from 8.77% to 96.49%. Three models with their hyperparameters and proportion of support vectors used below had the highest accuracy:

- Model 1 (26.59% of training data patients used for support vectors)
    - Poly PCA Kernel
    - 9 PCA Components
    - RBF SVM Kernel
    - No Slack Variable
- Model 2 (32.97% of training data patients used for support vectors)
- Poly PCA Kernel
    - Poly PCA Kernel
    - 7 PCA Components
    - Linear SVM Kernel
    - Linear Slack Variable,
- Model 3 (78.46% of training data patients used for support vectors)
    - Poly PCA Kernel
    - 4 PCA Components

- RBF SVM Kernel
- Linear Slack Variable

The tie breaker was determined by the model with the least number of PCA components to simplify the model per the simplicity principle of Occam's razor, so the third model with 4 PCA components was determined to be the best model which, when tested on the test set, had a slightly lower but still strong accuracy rate of 91%.

VII. Conclusion

From the models that were tested, the model with the strongest performance utilized a poly PCA kernel, 4 PCA components, an RBF SVM Kernel, and the linear slack variable. It utilized 78.46% of the examples in the training set and showed a 91% accuracy on the test data. From the model, utilizing a polynomial PCA kernel, which adds more dimensionality to the dataset past typical linear combinations of the features, leads to a stronger accuracy in the final validation and test dataset. Other parameters such as the slack variables, PCA components, and PCA Kernel are more dependent on what other hyperparameters were used in the model.

When utilizing this model, understanding the ramifications of incorrect diagnoses is extremely important. If the model creates a false positive, the risk of an invasive and unnecessary surgery could happen as a result. If the model is a false negative, the patient could go untreated which could be life threatening. Some ways to improve the model to minimize this is to increase the score used to determine what class they are in with an unidentifiable class becoming more common. It could also be used as a supplement with other tests to help increase the likelihood that the patient has a benign or malignant tumor. The model could also be optimized in other ways than what was done in the model such as other types of feature selection or other types of models such as neural networks or decision trees. Breast cancer is a very prevalent disease in our society. With new methods to determine if tumors found are benign or malignant, unnecessary surgeries can be prevented without risking patients getting the care they need when the tumor was first found.

VIII. References:

(1) "Breast Cancer Facts and Statistics" *Breast cancer facts and statistics*. [Online]. Available: https://www.breastcancer.org/facts-statistics.

(2) B. Krans, "Breast lump removal (lumpectomy): Risks, recovery, and more," *Healthline*, 21-Jul-2020. [Online]. Available: https://www.healthline.com/health/breast-lump-removal.

(3) K. H. Walker, *Clinical methods: The history, physical, and laboratory examinations. 3rd edition*, 3rd ed. LexisNexis UK, 1990.

(4) Y. M. H, "Breast cancer dataset," *Breast Cancer Dataset*, 29-Dec-2021. [Online]. Available: https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset.

(5) "Sklearn.model_selection.train_test_split," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. [Accessed: 30-Apr-2022].

(6) "RandomOversampler," *RandomOverSampler - Version 0.9.0*. [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html.

[7] A. Ng, "Part XI principal components analysis - Stanford University," *CS229 Lecture notes*. [Online]. Available: http://cs229.stanford.edu/notes2020spring/cs229-notes10.pdf.
[8] D. Leeds, Class Lecture, Topic: "Lecture 5 – Dimensionality Reduction." CISC5800L01, Fordham University, New York, NY., March 2022.
[9] "SKLEARN.DECOMPOSITION.KERNELPCA," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html.
[10] "Stat 241B / EECS 281B advanced topics in statistical learning theory," *EECS 281B / STAT 241B Home Page*. [Online]. Available: https://people.eecs.berkeley.edu/~wainwrig/stat241b/. [Accessed: 30-Apr-2022].
[11] "SKLEARN.DECOMPOSITION. PCA," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.
[12] D. Leeds, Class Lecture, Topic: "Lecture 4 – Support Vector Machine." CISC5800L01, Fordham University, New York, NY., February 2022.
[13] "NumPy user guide," *NumPy user guide - NumPy v1.22 Manual*. [Online]. Available: https://numpy.org/doc/stable/user/index.html.

IX. Appendix

Appendix A. Accuracy: PCA Components 1-5

Breast Cancer Model Comparison: Accuracy							
PCA Kernel	**Slack Variable**	**SVM Kernel**	**PCA_Components**				
			1	**2**	**3**	**4**	**5**
linear	**no**	linear	85.96%	84.21%	82.46%	84.21%	84.21%
linear	**no**	rbf	85.96%	91.23%	91.23%	89.47%	89.47%
linear	**linear**	linear	14.04%	14.04%	14.04%	10.53%	21.05%
linear	**linear**	rbf	14.04%	36.84%	42.11%	49.12%	45.61%
linear	**squared**	linear	14.04%	17.54%	36.84%	36.84%	42.11%
linear	**squared**	rbf	14.04%	8.77%	22.81%	36.84%	42.11%
poly	**no**	linear	78.95%	77.19%	92.98%	92.98%	89.47%
poly	**no**	rbf	85.96%	87.72%	91.23%	92.98%	92.98%
poly	**linear**	linear	77.19%	87.72%	92.98%	92.98%	89.47%
poly	**linear**	rbf	85.96%	31.58%	94.74%	96.49%	33.33%
poly	**squared**	linear	14.04%	0.00%	92.98%	92.98%	33.33%
poly	**squared**	rbf	14.04%	31.58%	36.84%	35.09%	31.58%
rbf	**no**	linear	68.42%	73.68%	73.68%	75.44%	75.44%
rbf	**no**	rbf	78.95%	77.19%	77.19%	78.95%	78.95%
rbf	**linear**	linear	15.79%	22.81%	22.81%	21.05%	22.81%
rbf	**linear**	rbf	15.79%	26.32%	24.56%	26.32%	31.58%
rbf	**squared**	linear	15.79%	15.79%	17.54%	17.54%	17.54%
rbf	**squared**	rbf	15.79%	15.79%	17.54%	17.54%	17.54%
sigmoid	**no**	linear	70.18%	77.19%	78.95%	80.70%	78.95%

sigmoid	**no**	rbf	70.18%	82.46%	82.46%	82.46%	82.46%
sigmoid	**linear**	linear	12.28%	45.61%	43.86%	42.11%	38.60%
sigmoid	**linear**	rbf	14.04%	45.61%	43.86%	45.61%	42.11%
sigmoid	**squared**	linear	14.04%	43.86%	17.54%	17.54%	17.54%
sigmoid	**squared**	rbf	14.04%	45.61%	19.30%	17.54%	19.30%

Note: Models highlighted in yellow highlighted had the highest accuracy rates, model highlighted in red was not a successful model since all alphas of one class went to 0.

Appendix B. Accuracy: PCA Components 6-10

Breast Cancer Model Comparison: Accuracy							
PCA Kernel	**Slack Variable**	**SVM Kernel**	**PCA Components**				
			6	**7**	**8**	**9**	**10**
linear	**no**	linear	84.21%	84.21%	84.21%	84.21%	84.21%
linear	**no**	rbf	87.72%	87.72%	87.72%	87.72%	87.72%
linear	**linear**	linear	17.54%	19.30%	21.05%	21.05%	21.05%
linear	**linear**	rbf	50.88%	52.63%	75.44%	77.19%	77.19%
linear	**squared**	linear	49.12%	49.12%	43.86%	45.61%	43.86%
linear	**squared**	rbf	49.12%	49.12%	43.86%	45.61%	43.86%
poly	**no**	linear	94.74%	94.74%	92.98%	96.49%	87.72%
poly	**no**	rbf	92.98%	92.98%	92.98%	92.98%	92.98%
poly	**linear**	linear	91.23%	96.49%	68.42%	70.18%	57.89%
poly	**linear**	rbf	35.09%	40.35%	45.61%	47.37%	66.67%
poly	**squared**	linear	26.32%	33.33%	29.82%	29.82%	29.82%
poly	**squared**	rbf	31.58%	33.33%	36.84%	35.09%	31.58%
rbf	**no**	linear	75.44%	75.44%	77.19%	75.44%	75.44%
rbf	**no**	rbf	82.46%	80.70%	82.46%	80.70%	80.70%

rbf	**linear**	linear	24.56 %	26.32 %	26.32 %	26.32 %	26.32 %
rbf	**linear**	rbf	33.33 %	33.33 %	35.09 %	35.09 %	35.09 %
rbf	**squared**	linear	21.05 %	21.05 %	21.05 %	21.05 %	21.05 %
rbf	**squared**	rbf	21.05 %	21.05 %	21.05 %	21.05 %	21.05 %
sigmoid	**no**	linear	80.70 %	80.70 %	80.70 %	80.70 %	80.70 %
sigmoid	**no**	rbf	87.72 %	87.72 %	85.96 %	85.96 %	85.96 %
sigmoid	**linear**	linear	43.86 %	43.86 %	43.86 %	43.86 %	43.86 %
sigmoid	**linear**	rbf	49.12 %	49.12 %	49.12 %	47.37 %	47.37 %
sigmoid	**squared**	linear	26.32 %	22.81 %	24.56 %	24.56 %	24.56 %
sigmoid	**squared**	rbf	26.32 %	28.07 %	28.07 %	28.07 %	28.07 %

Note: Models highlighted in yellow highlighted had the highest accuracy rates.

Appendix C. Percent of Examples for Support Vectors: Components 1-5

Breast Cancer Model Comparison: Percent of Examples for Support Vectors							
PCA Kernel	**Slack Variable**	**SVM Kernel**	**PCA Components**				
			1	**2**	**3**	**4**	**5**
linear	**no**	linear	100.00 %	72.75 %	71.65 %	70.77 %	70.77 %
linear	**no**	rbf	83.30%	71.87 %	78.46 %	91.43 %	96.48 %
linear	**linear**	linear	5.49%	1.98%	2.64%	3.08%	2.20%

linear	**linear**	rbf	7.03%	29.89 %	31.43 %	43.08 %	38.02 %
linear	**squared**	linear	0.44%	3.52%	3.52%	2.20%	3.08%
linear	**squared**	rbf	5.49%	19.34 %	16.26 %	14.95 %	13.85 %
poly	**no**	linear	81.76%	70.11 %	34.51 %	32.75 %	31.65 %
poly	**no**	rbf	93.19%	90.55 %	73.85 %	71.87 %	68.35 %
poly	**linear**	linear	87.25%	25.71 %	41.76 %	37.14 %	39.78 %
poly	**linear**	rbf	89.23%	11.21 %	78.02 %	78.46 %	18.90 %
poly	**squared**	linear	6.37%	0.00%	50.11 %	45.71 %	5.05%
poly	**squared**	rbf	7.25%	7.91%	12.75 %	11.65 %	7.69%
rbf	**no**	linear	60.22%	56.26 %	56.48 %	56.48 %	56.48 %
rbf	**no**	rbf	60.00%	56.26 %	57.36 %	59.12 %	58.90 %
rbf	**linear**	linear	14.51%	26.81 %	26.37 %	26.59 %	26.15 %
rbf	**linear**	rbf	24.62%	25.49 %	26.15 %	26.81 %	29.01 %
rbf	**squared**	linear	11.65%	10.11 %	10.99 %	11.21 %	12.09 %
rbf	**squared**	rbf	11.87%	10.77 %	13.41 %	13.85 %	12.97 %
sigmoid	**no**	linear	59.12%	55.60 %	54.29 %	53.85 %	54.73 %
sigmoid	**no**	rbf	59.12%	54.29 %	54.95 %	54.95 %	56.26 %
sigmoid	**linear**	linear	12.53%	38.24 %	36.92 %	36.04 %	37.58 %
sigmoid	**linear**	rbf	13.85%	41.76 %	40.44 %	40.00 %	40.66 %
sigmoid	**squared**	linear	10.11%	41.98 %	14.51 %	13.85 %	11.21 %
sigmoid	**squared**	rbf	10.11%	40.66 %	15.82 %	14.95 %	17.36 %

Note: Models highlighted in yellow highlighted had the highest accuracy rates, model highlighted in red was not a successful model since all alphas of one class went to 0.

Appendix D. Percent of Examples for Support Vectors: Components 6-10

Breast Cancer Model Comparison: Percent of Examples for Support Vectors							
PCA Kernel	**Slack Variable**	**SVM Kernel**	**PCA Components**				
			6	**7**	**8**	**9**	**10**
linear	**no**	linear	70.77 %	70.99 %	70.77 %	70.77%	70.77%
linear	**no**	rbf	98.46 %	99.34 %	99.78 %	100.00 %	100.00 %
linear	**linear**	linear	2.42%	3.52%	4.18%	3.74%	2.86%
linear	**linear**	rbf	46.59 %	48.57 %	65.49 %	68.13%	70.55%
linear	**squared**	linear	3.52%	3.52%	3.52%	3.52%	3.52%
linear	**squared**	rbf	13.85 %	13.85 %	14.29 %	13.63%	13.41%
poly	**no**	linear	26.81 %	27.03 %	27.25 %	26.59%	26.81%
poly	**no**	rbf	45.49 %	49.01 %	51.43 %	52.75%	49.01%
poly	**linear**	linear	32.75 %	32.97 %	14.95 %	16.26%	10.77%
poly	**linear**	rbf	20.66 %	22.64 %	26.59 %	26.15%	41.98%
poly	**squared**	linear	5.93%	2.20%	0.88%	1.10%	0.88%
poly	**squared**	rbf	8.57%	10.99 %	10.77 %	11.43%	8.35%
rbf	**no**	linear	55.16 %	55.16 %	55.82 %	54.51%	54.51%
rbf	**no**	rbf	59.34 %	61.32 %	61.10 %	61.32%	61.10%
rbf	**linear**	linear	26.59 %	27.47 %	27.25 %	28.79%	27.69%
rbf	**linear**	rbf	28.57 %	31.43 %	32.09 %	32.53%	32.09%
rbf	**squared**	linear	10.55 %	10.99 %	11.43 %	11.21%	11.65%

rbf	**squared**	rbf	13.41 %	14.29 %	12.09 %	13.63%	13.85%
sigmoid	**no**	linear	53.19 %	53.19 %	53.19 %	53.19%	53.19%
sigmoid	**no**	rbf	56.26 %	56.26 %	56.26 %	56.04%	56.26%
sigmoid	**linear**	linear	38.90 %	38.68 %	38.24 %	38.02%	38.90%
sigmoid	**linear**	rbf	40.88 %	42.42 %	42.64 %	42.64%	42.86%
sigmoid	**squared**	linear	14.29 %	15.38 %	15.60 %	15.38%	14.51%
sigmoid	**squared**	rbf	20.44 %	21.32 %	21.32 %	21.54%	20.88%

Note: Models highlighted in yellow highlighted had the highest accuracy rates.