654: Machine Learning Final Report

Havisha Khurana

Material for the Final Project

The code for the final project is spread over the following Kaggle notebooks.

1. Read image data and extract features for 50,000 images. here

2. Reduce features by removing zero variance, near zero variance, and highly correlated features from the training data set. here

3. Logistic regression models on large subset (49921 rows and 1373 columns). here

4. Logistic regression models on small subset with all features (5004 rows and 8194 columns). here

5. Other models on small subset by preserving those features that had high importance on logistic regression models.

Note: The original image data was accessed from Kaggle. See here

Research problem

The project focuses on an image classification task, precisely whether cancer was found or not in medical images of regions of the breast.Cancer is the second leading cause of deaths among American women of all races and origin (Heron 2021), and the leading cause of death in the world (Ferlay et al. 2021). Furthermore, breast cancer is among the most common forms of cancer, accounting for 10 million cases worldwide in 2020 (Ferlay et al. 2021). There are two primary types of breast cancers, Ductal carcinoma in situ (DCIS) and Invasive breast cancer (ILS or IDC), and the latter makes up for about 70-80% of all such cases (Sharma et al. 2010). The diagnosis of the same is time consuming and challenging primarily because pathologist have to scan a large swathes of benign regions to ultimately identify the areas of malignancy (Janowczyk and Madabhushi 2016; Cruz-Roa et al. 2014). Precise identification of IDC is required to direct patient treatment. In recent years, many machine learning models using deep neural networks have been used to aid medical and biological classification (Janowczyk and Madabhushi 2016; Cruz-Roa et al. 2014). Thus, one potential benefit of a predictive model for this outcome is to speed-up the diagnosis and begin treatment for patients with this condition.

Another personal motivation to pursue this project is to get exposure working with image data. I anticipate the following challenges while working with the image data set. First, each image is a large file and requires extensive computational resources. Therefore, I can not load all the images and transform it at once. Second, the quality of an image is contingent on many factors, eg., the device from which it's taken, the magnification, the angle, blur, zoom, flip, etc. Therefore, two images that appear the same to human eye, might be very viewed differently by the computer. To overcome this challenge, data augmentation is performed at times to add random noise to the training dataset for replicating close to real-life scenarios. However, this would increase the sample size, which further burdens the computational resources. In this project, however, since I'll be working with medical images taken by professional radiologists, I do not expect only this to be a major roadblock. Lastly, I am working with colored images, therefore, each image will be represented by a three-dimensional array, with each cell representing a pixel. Though, in my professional life, I don't anticipate working with medical or biological images, I believe this is a good first step to building skills for image analysis using machine learning.

Description of the data

The description of the data is divided into the following subsections: data source, data representation, feature extraction, and data reduction.

Data Source

The data comes from a 2014 study (Cruz-Roa et al. 2014), with images of the entire region of the breast for 162 cancer patients. I accessed it through Kaggle (link provided in the beginning of the report). It consists of whole mount slide images of Breast Cancer specimens scanned at 40x. Each slide, covering the entire breast area, was broken into 277,524 patches of size 50 x 50. Of the total, 198,738 (71.5%) patches were IDC negative and 78,786 (28.5%) patches were IDC positive.

Data Representation

The data is provided in .jpg format. I used Python module Keras [@] to preprocess the images. First, I read the .jpg image in the environment. Next, I converted each image to an array. As a result, each patch of image is a 3-D array of 50 x 50, where each value in the matrix represents the pixel brightness on the the particular row and column combination. The dimensions represents the red, blue, and green light channels (RGB). Figure 1 shows how an image looks in light different channels.

(a) Image in RGB channel

(c) Image in Red channel

(b) Image in Blue Channel
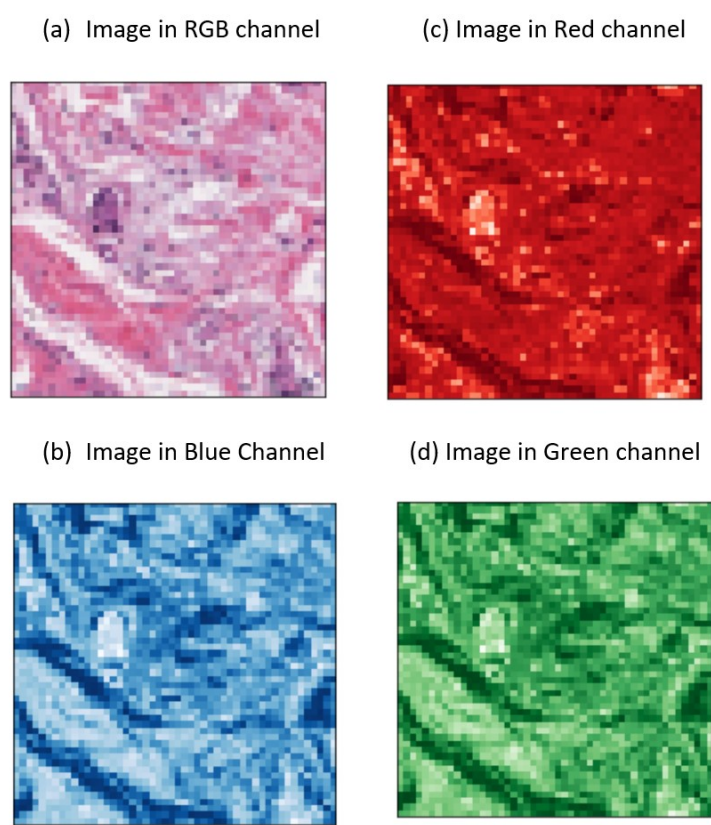
(d) Image in Green channel

Figure 1: Image in RGB

Figure 2 shows 5 random images in RGB and whether the patch was cancerous or not.
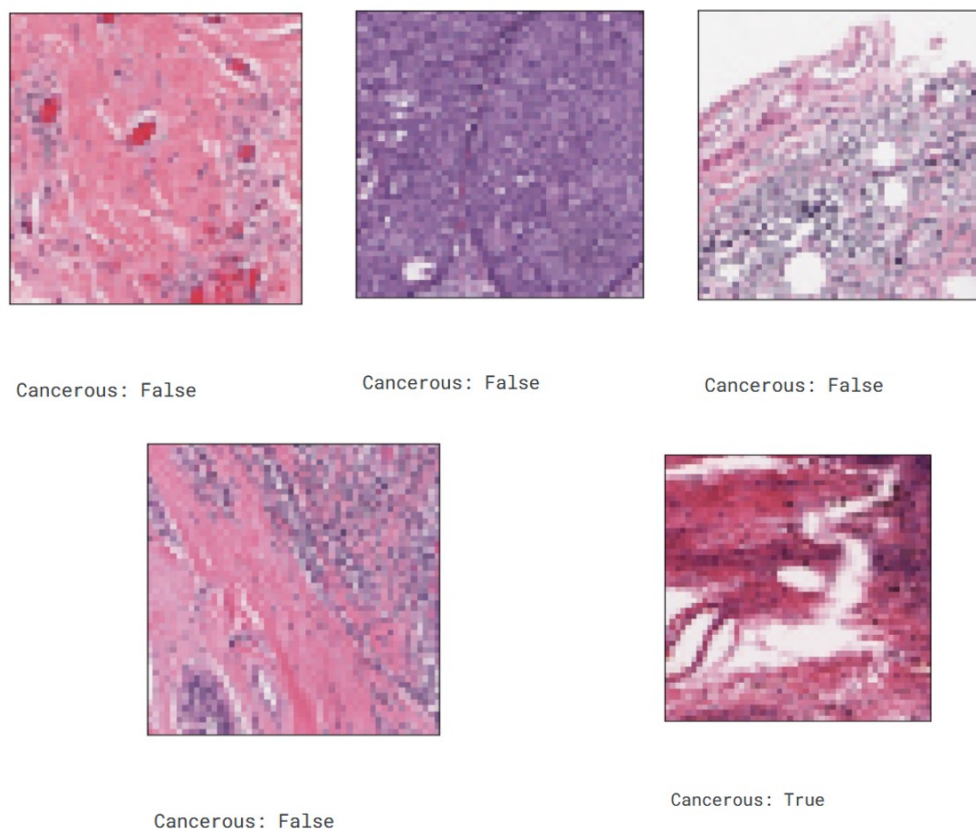

Figure 2: Random Sample of 5 images

Therefore, each image file can be associated with 7,500 (50 x 50 x 3) numbers representing the pixel brightness in the three light channels. However, these number are not meaningful in themselves, as they don't communicate how the image appears as a whole. There are several ways image features can be calculated, for instance, the mean intensity of image in all channels, the edges and boundaries in an image, number of elements in an image, etc. Since it requires expertise to know which features to extract, I decided not to hand-define any custom features. Instead, I'll use a pre-trained image classification model, ResNet-50, which generate 8,192 features for the images in this data set.

Data Features using ResNet-50

ResNet-50, short for Residual Networks, is a convolutional neural network that is 50 layers deep. It has been pre-trained on more than a million images from the ImageNet database, and one if its more complex architecture won the ImageNet competition in 2015 (He et al. (2016)). The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. The network architecture is shown in Figure 3.

I used Python module, Keras(Chollet 2015) and TensorFlow(Abadi et al. 2015), to load the pre-trained model with original training weights and extracted 8,142 features from each image.
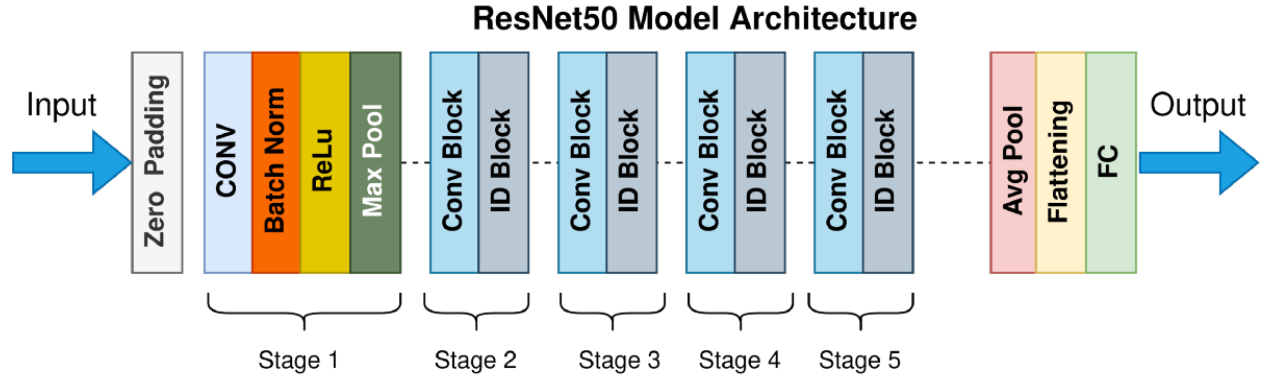
Figure 3: ResNet-50 architecture

Data Reduction

Given the very large size of the original dataset and the limited computations power on my end, I decided to work with two subsets samples of this data, one relatively large sample with close to 50,000 random images, and other, a smaller sample with 5,000 images. The smaller sample was taken from the larger sample.

The larger sample is a random selection of close to 50,000 images. The dataset with all 8,000 features was 3Gb large and took up 20Gb RAM on opening. Since this would have drastically limited the resources for model training, I decided to reduce the features for this sample. First, I split the data into training and testing datasets. Next, on the training dataset, I removed all features that had zero variance, near zero variance, or a correlation of 0.5. This reduced the number of features to 1371. Thus, the larger sample had a final size of 49,921 rows and 1,373 columns.

The smaller sample had all the 8,192 features. However, the recipe included steps to remove columns based on the same criteria as mentioned above. This dataset had a final size of 5,004 rows and 8,194 columns. Note there was one ID column and one outcome column in both the datasets.

Description of the models

Since this is a classification problem with two classes, positive or negative for IDC, I will use logistic regression. Given the large number of features, over-fitting is possible, and so I will run penalized logistic regressions as well with both ridge penalty and lasso penalty. Logistic regression is analogous to linear regression, however, instead of predicting a continuous outcome, it predicts the probability of one of the two outcome classes. I will choose models based on the logistic loss criteria, i.e., a model with the small logistic loss. The model assumes that the predictors have a linear relationship with the outcome, all important predictors are present in the model, and there is no multicollinearity.In the regularized version of the logistic regression, we penalize the model for over-fitting by adding a penalty term. This constrains the size of the predicted coefficients.

In unregularized logistic regression, there are no hyperparameter to be tuned. In ridge regression, the penalty is the euclidean norm, whereas in lasso regression, the penalty is the least absolute shrinkage. Thus, there are on hyperparameter each in ridge and lasso regressions to tune.

Implementation Plan

First, split the sample into training and test set, with an 80-20 ratio. Next, make indices for 10-fold cross validation. Then, use recipe package, and define the first column (serial_num) as an ID, the second column (truth_val) as an outcome, and the remaining columns as predictors. Next, remove removes any variable

with zero variance or near-zero variance, and keep only one of the columns which are highly correlated (a correlation of 0.5 or more). Lastly, transform the outcome into 1 and 2 instead of 0 and 1, to avoid a run-time error.

Metric for model comparison

For model comparison, I will use area under the receiver operating characteristic (ROC) curve (AUC) as the metric. I will also compute the confusion metric, and evaluate the true positive rate, true negative rate, accuracy, and precision.

Packages required

I used the following Python and R packages for the project: pandas (team 2020), NumPy (Harris et al. 2020), keras (Chollet 2015), scikit-leanr (Pedregosa et al. 2011), TensorFlow (Abadi et al. 2015), Matplotlib (Hunter 2007), tidyverse (Wickham et al. 2019), recipes (Kuhn and Wickham 2022), caret (Kuhn 2022), base R (R Core Team 2021), cutpointr(Thiele and Hirschfeld 2021), vip(Greenwell and Boehmke 2020), glmnet(Friedman, Hastie, and Tibshirani 2010).

Model Fit

I will discuss the model results for the small and large samples. While discussion the results, I will share the metrics for model comparison, the classification plot, as well as the logistic loss plot for the hyperparamters.

Model Fit for small sample

Table 1 summarizes the model fit on the small sample for the three models. The logLoss for unregularized, regularized with ridge penalty, and regularized with lasso penalty were 0.371, 0.342, and 0.357 respectively. The area under the ROC curve for the three models were 0.914, 0.921, and 0.919. As was expected, regularization improved the results slightly, as this sample had relatively large features to observation ratio. The best tune hyperparameters for the ridge logistic regression was 0.066, and for lasso losgistic regression was 0.0038. On this smaller sample, I choose the regularized logistic regression with ridge penalty as the final model for it's slight improvement. I chose a cut-off of 0.297 for this model. This cut-off was found found using the cutpointr package (Thiele and Hirschfeld 2021), and maximizes the distinction between the negative and positive sets.
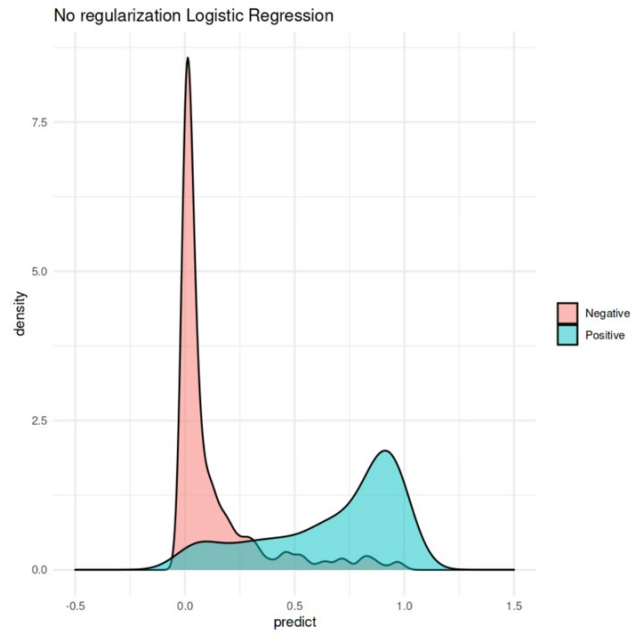
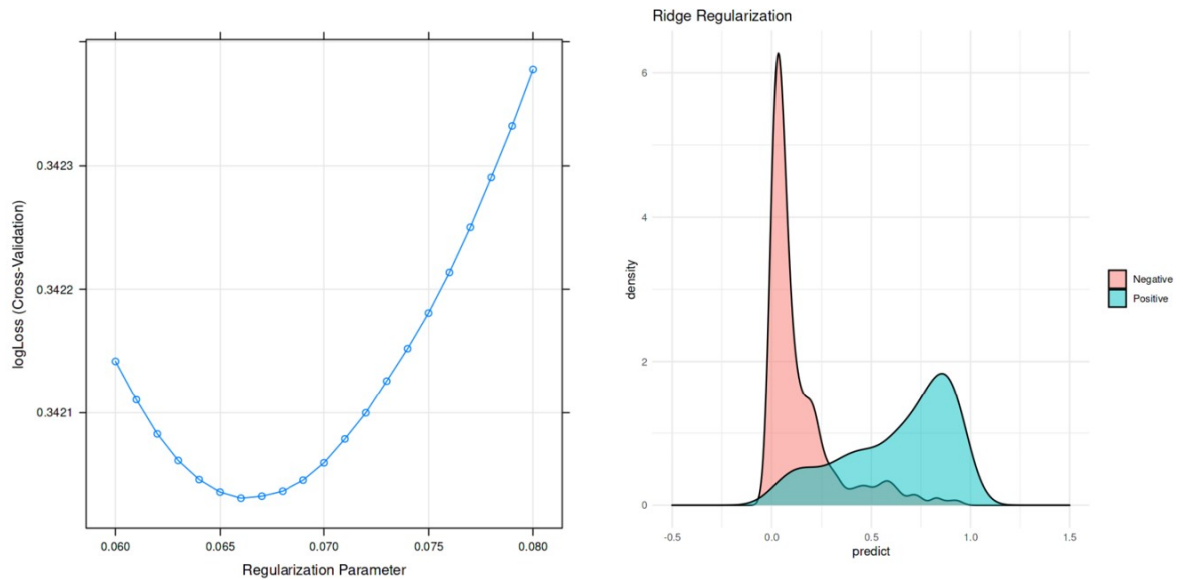Figure 4: Classification Plot for unregularized logistic regression plot

Figure 5: Regularization paramter and classification plot for regularized logistic regression (ridge penalty)
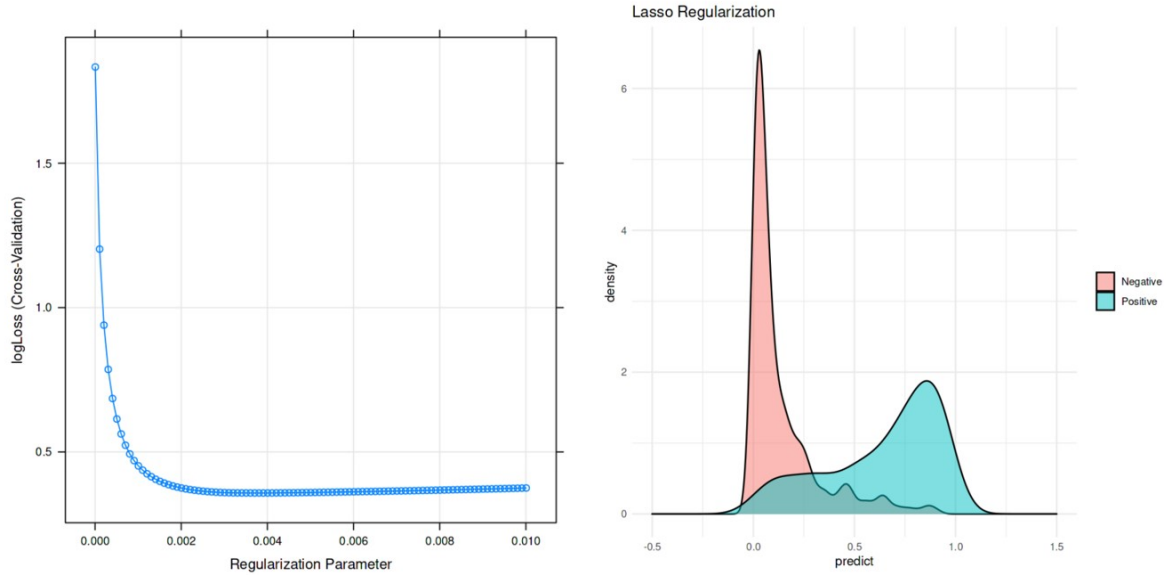
Lasso Regularization

Table 1

Model performance for small sample

Model	logLoss	Cut-off	AUC	Accuracy	True Positive rate	True Negative rate	Precision
Unregularized Logistic Regression	0.371	0.286	0.914	0.858	0.849	0.86	0.686
Logistic Regression with Ridge Penalty	0.342	0.297	0.921	0.867	0.857	0.87	0.703
Logistic Regression with Lasso Penalty	0.357	0.308	0.919	0.861	0.834	0.87	0.697

Model Fit for large sample

Table 2 summarizes the model fit on the small sample for the three models. The logLoss for unregularized, regularized with ridge penalty, and regularized with lasso penalty were 0.369, 0.368, and 0.369 respectively. The area under the ROC curve for the three models was 0.897 in all cases. As was expected, all regressions performed alike becuase of the smaller features to observations ratio. The best tune hyperparameters for the ridge logistic regression was 0.031, and for lasso losgistic regression was 0.001. On this larger sample, I choose the unregularized logistic regression for slightly higher precision. I chose a cut-off of 0.283 for this model. This cut-off was found found using the cutpointr package (Thiele and Hirschfeld 2021), and maximizes the distinction between the negative and positive sets.
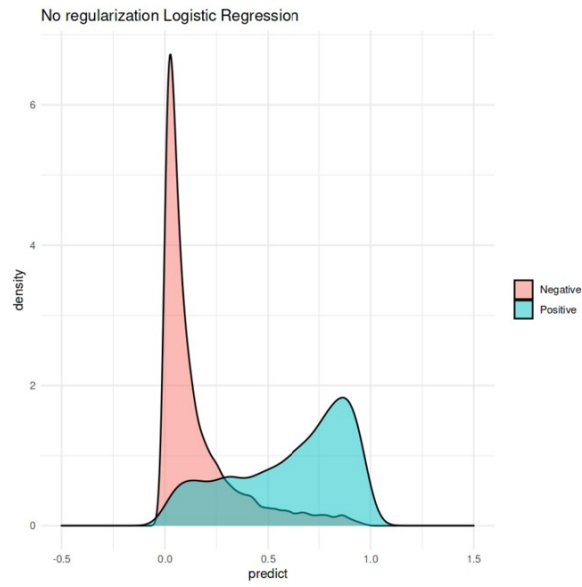
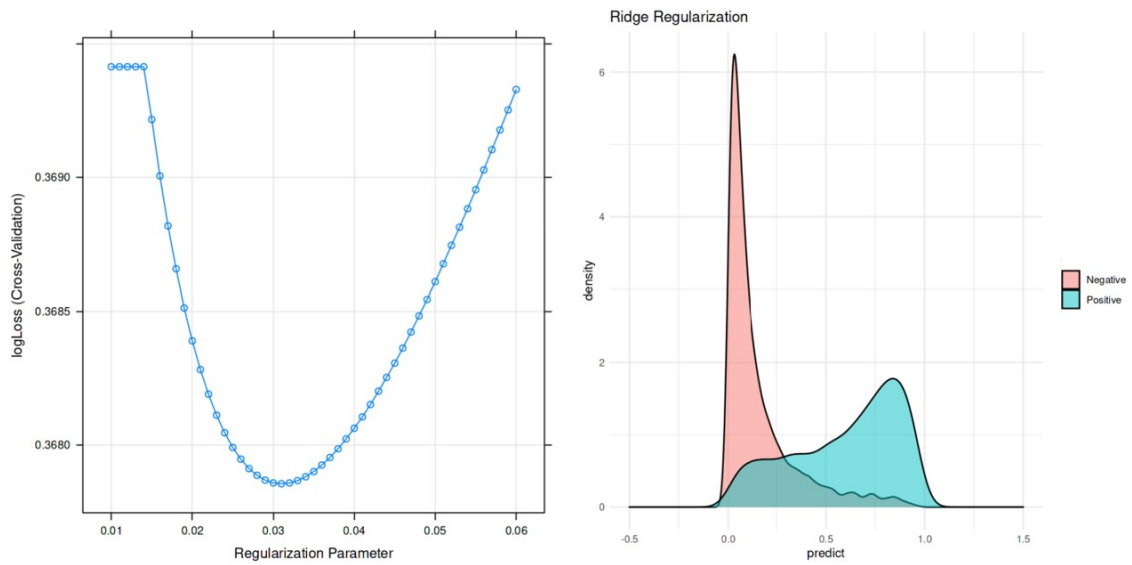Figure 6: Classification Plot for unregularized logistic regression plot

Figure 7: Regularization paramter and classification plot for regularized logistic regression (ridge penalty)
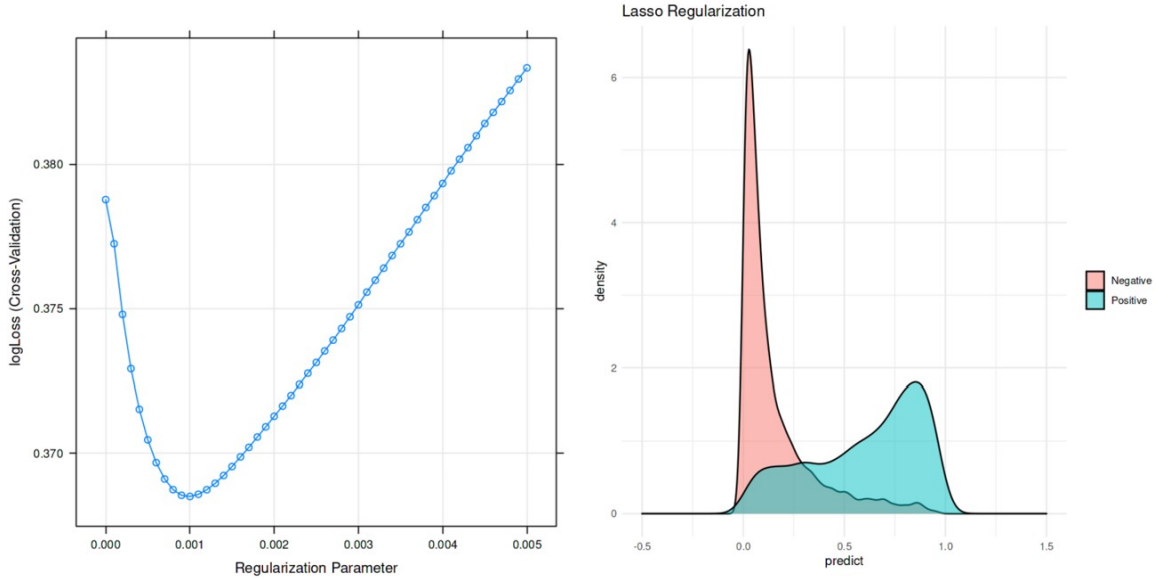
Figure 8: Regularization paramter and classification plot for regularized logistic regression (lasso penalty)

Table 2

Model performance for large sample

Model	logLoss	Cut-off	AUC	Accuracy	True Positive rate	True Negative rate	Precision
Unregularized Logistic Regression	0.369	0.283	0.897	0.83	0.824	0.832	0.657
Logistic Regression with Ridge Penalty	0.368	0.275	0.897	0.827	0.832	0.825	0.65
Logistic Regression with Lasso Penalty	0.36	0.274	0.897	0.826	0.832	0.823	0.647

Discussion

All the three logistic models performed similarly, and the AUC was around 0.9 in all models and samples alike. This indicates that regularization doesn't significantly improve results. One reason for this could be to reduce the number of features at the outset, that reduced the possibility of potential over-fit. In the smaller sample, however, there were small improvements with regularization (a 0.1 improvement in the AUC metric). This could be because there were about 1,000 features for a training set of 4,000, i.e., a features to observation ratio of 1:4. In the larger sample, there was no difference between the models at all. Here, there were again 1,300 features but 40,000 observation, a feature to observation ratio of close to 1:40. However, the model did well and had an accuracy of around 83-86%. I would say I was surprised how well the model performed with the ResNet-50 features, however, expected the different models to perform in the way they did due to the sizes. The model performance slightly dropped on the larger set, this could probably be because of the larger sample size.

10

There were around 20 ResNet-50 features that were assigned an importance of more than 50, and 700 features with importance of more than 20. However, just reading the features wasn't very insightful by itself. There are a number of limitations of this project. First, the feature-generating model, ResNet-50, was not developed

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." https://www.tensorflow.org/.

Chollet, François. 2015. "Keras." *GitHub Repository.* https://github.com/fchollet/keras; GitHub.

Cruz-Roa, Angel, Ajay Basavanhally, Fabio González, Hannah Gilmore, Michael Feldman, Shridar Ganesan, Natalie Shih, John Tomaszewski, and Anant Madabhushi. 2014. "Automatic Detection of Invasive Ductal Carcinoma in Whole Slide Images with Convolutional Neural Networks." In *Medical Imaging 2014: Digital Pathology*, 9041:904103. SPIE.

Ferlay, Jacques, Murielle Colombet, Isabelle Soerjomataram, Donald M Parkin, Marion Piñeros, Ariana Znaor, and Freddie Bray. 2021. "Cancer Statistics for the Year 2020: An Overview." *International Journal of Cancer* 149 (4): 778–89.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2010. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1): 1–22. https://doi.org/10.18637/jss.v033.i01.

Greenwell, Brandon M., and Bradley C. Boehmke. 2020. "Variable Importance Plots—an Introduction to the Vip Package." *The R Journal* 12 (1): 343–66. https://doi.org/10.32614/RJ-2020-013.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–78.

Heron, Melonie. 2021. "National Vital Statistics Reports." *National Vital Statistics Reports* 70 (3).

Hunter, J. D. 2007. "Matplotlib: A 2d Graphics Environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Janowczyk, Andrew, and Anant Madabhushi. 2016. "Deep Learning for Digital Pathology Image Analysis: A Comprehensive Tutorial with Selected Use Cases." *Journal of Pathology Informatics* 7 (1): 29.

Kuhn, Max. 2022. *Caret: Classification and Regression Training.* https://CRAN.R-project.org/package=caret.

Kuhn, Max, and Hadley Wickham. 2022. *Recipes: Preprocessing and Feature Engineering Steps for Modeling.* https://CRAN.R-project.org/package=recipes.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Sharma, Ganesh N, Rahul Dave, Jyotsana Sanadya, Piush Sharma, KK3255438 Sharma, et al. 2010. "Various Types and Management of Breast Cancer: An Overview." *Journal of Advanced Pharmaceutical Technology & Research* 1 (2): 109.

team, The pandas development. 2020. *Pandas-Dev/Pandas: Pandas* (version latest). Zenodo. https://doi.org/10.5281/zenodo.3509134.

Thiele, Christian, and Gerrit Hirschfeld. 2021. "cutpointr: Improved Estimation and Validation of Optimal Cutpoints in R." *Journal of Statistical Software* 98 (11): 1–27. https://doi.org/10.18637/jss.v098.i11.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.