Dennis Shen
UID: 304755705
3/18/21
BE C275 - Professor A. Meyer
Body Word Count: 1439

Developing Classifiers for Diabetic Retinopathy

Introduction

Diabetic retinopathy (DR) is a complication that can arise in people that have diabetes mellitus and can lead to a variety of sight related issues including but not limited to blurry vision, loss in color perception, poorer night vision, and even blindness in severe cases [1]. DR arises when elevated blood sugar levels damage blood vessels in the retina. For those living with diabetes, proper eye exams and diagnosis of a potential onset of DR is critical to mitigating any severe complications, however, diagnosis of DR requires time, expertise, and money.

A whole hardware and software ecosystem could be built around easily diagnosing DR, which would assist ophthalmologists and give those with diabetes a way to better monitor their own health.

Problem Definition

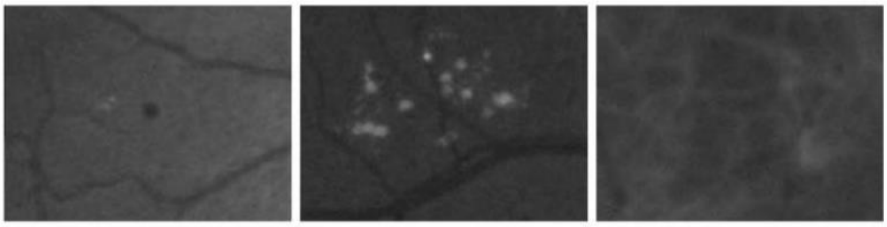


Figure 1: Image taken from Antal and Hajdu (2014), showcasing some of the signs of DR from patient imaging data [2].

From Antal and Hajdu's paper, 366 million people had diabetes worldwide in 2011, with an additional 280 million at risk. Of the 366 million with diabetes, 40% suffered from DR complications [2]. With the multiple tools that machine learning provides for developing models and analyzing data, an automated classifier for diagnosing DR based on data such as the presence of microaneurysms or anatomical data would lay the groundwork for an increased level of healthcare for those with diabetes. With so many cases of DR worldwide, there is a lot of data that could potentially be used to provide insight into how to more accurately and rapidly diagnose DR. One key topic that this project will address is the viability of using classifiers such as logistic regression, KNN, or SVM to predict the presence of DR, and if any of the models are better than others in diagnosing DR.

Methods

Algorithm Description & Rationale

Antal and Hajdu's original work was to automatically extract images and use an ensemble based approach to detect DR, with hopes of eventually applying the model to new test images [2]. For this project, Antal and Hajdu's original work was used as inspiration for developing a less complicated classifier, with a reliance on a smaller subset of their data.

For this project, the problem of developing a DR classifier falls under the scope of supervised learning - classification. A flow chart is attached outlining the exact process that was used to develop each model, with additional comments in the project code to explain any complicated features or choices.

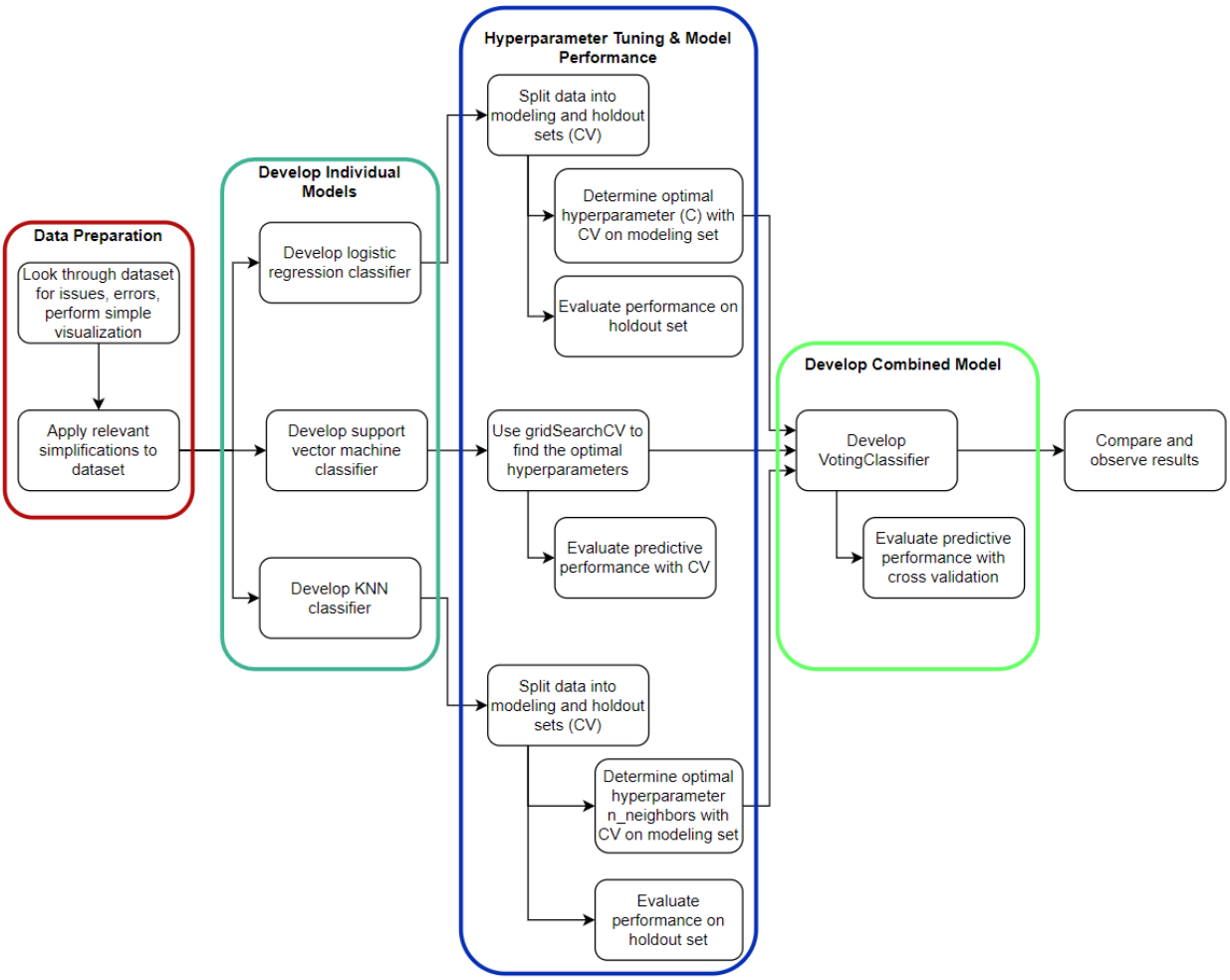


Figure 2: Flowchart used to guide classifier development.

The first step in this project was to scrub through the dataset provided, ensure that there were no errors, and see if there were any changes that needed to be made.

For each of the models, their hyperparameters were tuned to a reasonable degree in order to extract the best predictive performance.

- In logistic regression, the default penalty is the L2 penalty with regularization applied by default. The C hyperparameter represents the inverse of the regularization strength, where smaller C values lead to stronger regularization [3].
- In SVMs, there are a multitude of parameters that can be tuned, including the kernel, C value (regularization parameter), gamma value (kernel coefficient), degree for the polynomial kernel function, and so on [4]. These were tuned using `gridSearchCV`, which provides the best parameters to use with the model based on a specific metric.
- Finally, in KNN, the hyperparameter that was tuned was the number of neighbors, or n_neighbors [5].

From the documentation, the `VotingClassifier` method takes a combination of different models and compares the output of each classifier. If a majority of the classifiers vote for a given class, then that sample is labeled that class. This was used in conjunction with the tuned logistic regression, SVM, and KNN models from earlier.

Since there was hyperparameter tuning, nested cross validation was used in order to develop a holdout set, while the model was tuned based on the modeling set. The data was fir split into modeling and holdout with 5 fold CV, then the modeling set was split up into 10 fold CV for hyperparameter tuning. The feature data was scaled using the `StandardScaler` before being used for any fitting or model development, in order to avoid cross contamination.

Development and Project Usage

The following packages (with version numbers) were used in this project:

- `sklearn (0.24.1)`
- `pandas (1.2.3)`
- `numpy (1.20.1)`
- `seaborn (0.11.1)`
- `matplotlib (3.3.4)`
- `scipy (1.6.1)`

In order to run all of the different models and visualizations, the user will need the dataset from the UCI Machine Learning Repository and preferably `Python 3.7.9 (64-bit)`, which is what this project was developed on.

Initial Data Visualization

From Antal and Hajdu's original paper, data was provided that spanned three categories: image-level, lesions, and anatomical. However, with the provided data there were two sets of features that had high collinearity, captured in variables 2 to 7 and 8 to 15. This can be observed in the heatmaps, where the two large regions of yellow rectangles capture the highly correlated features.

From the dataset documentation, these features describe the number of microaneurysms and exudates pixels at varying confidence levels, from low to high, respectively. In order to simplify the data, only the feature sets with the highest confidence levels were used.

The next step was to try to visualize the data and see if there were any obvious decision boundaries that could be made, but there was no luck in this method, as shown below.

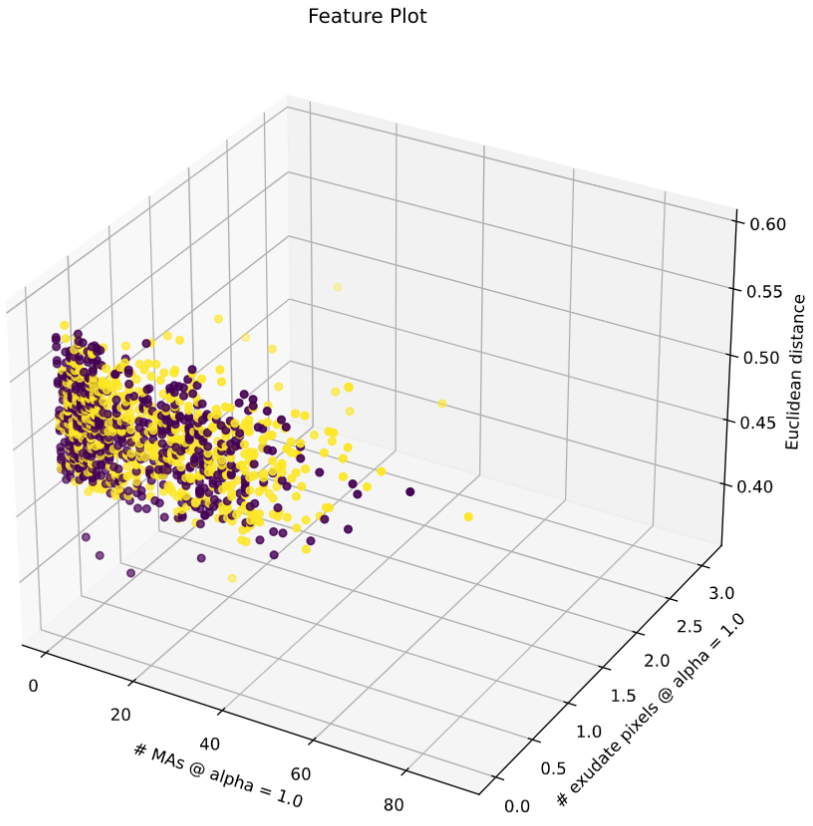


Figure 3: 3D Plot comparing number of microaneurysms, exudate pixels, and Euclidean distance of the centers of the macula and optic discs.

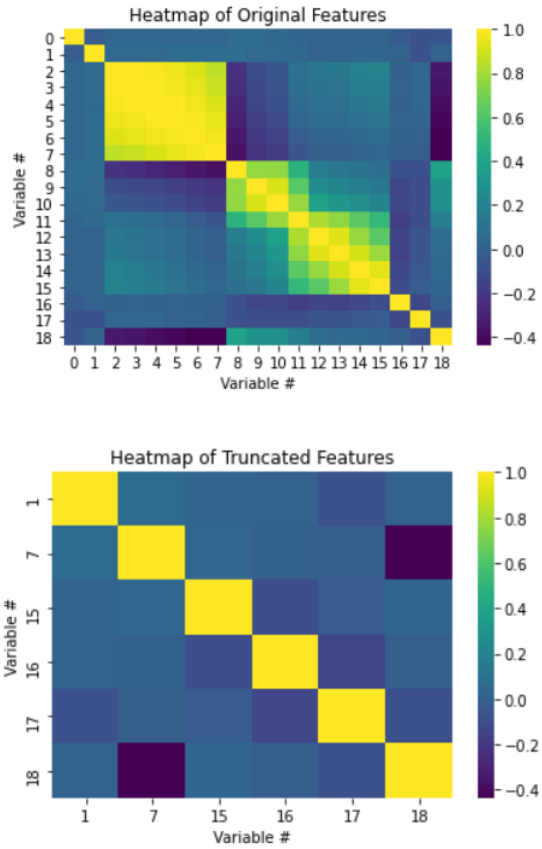


Figure 4: Heatmap of data before and after truncation.

Results

With the framework for the project laid out, each of the individual models were developed. By comparing each of the different classification methods, they all have relatively similar performance, but with notably the combined classifier having the highest accuracy. For the tuning results, the graphs are available on the notebook or in the *Appendix* of this report.

Classifier:	**Tuned Hyperparameter:**	**Accuracy:**	**Sensitivity / Specificity:**
Logistic Regression	C: 4.371	59.51%	62.52% / 56.11%
Support Vector Machine	Kernel: RBF, C: 1000, gamma: 0.01	60.73%	53.35% / 69.07%
K-Nearest Neighbors	N_neighbors: 210	57.43%	62.68% / 51.48%
Combined	N/A	61.16%	58.26% / 64.44%

Table 1: Summary of model results.

Logistic Regression

The best model for logistic regression was developed when using a larger value of C, which means that there was less regularization applied to the model. This served as a good baseline to compare the other models against.

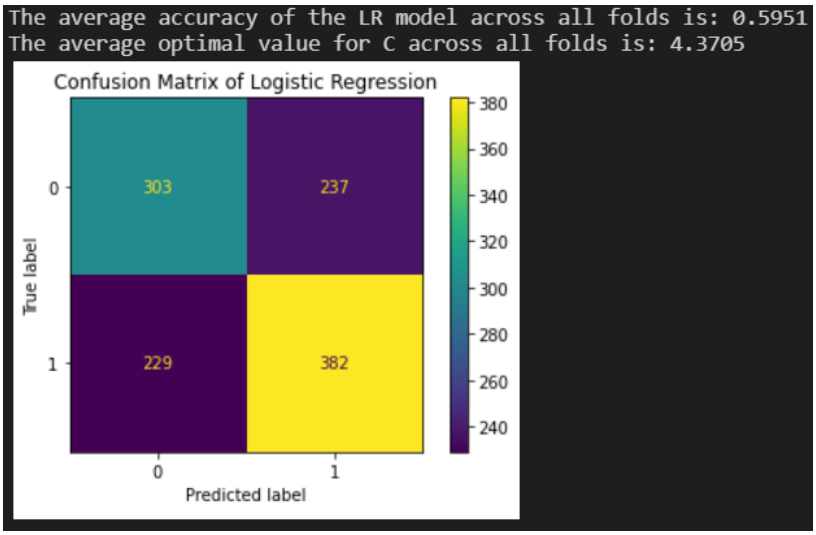


Figure 5: Logistic regression results and confusion matrix (CM).

Support Vector Machine

In general, the SVM performed well compared to the other two methods, but could probably use even more tuning in order to produce an even more accurate classifier.

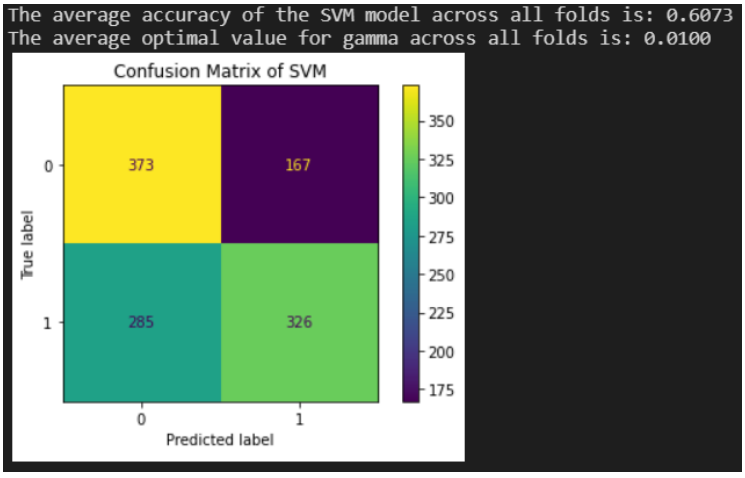


Figure 6: SVM (RBF kernel) results and CM. When tuning the support vector machine, the first step was completed using gridSearchCV to tune the kernel type and regularization parameter C. The gamma value was not included in this search as it refused to spit out a result (even past 45 minutes running on a 6-core CPU).

KNN Classifier

The KNN classifier performed the worst in terms of accuracy out of the classifiers but did have the highest relative sensitivity out of all classifiers.

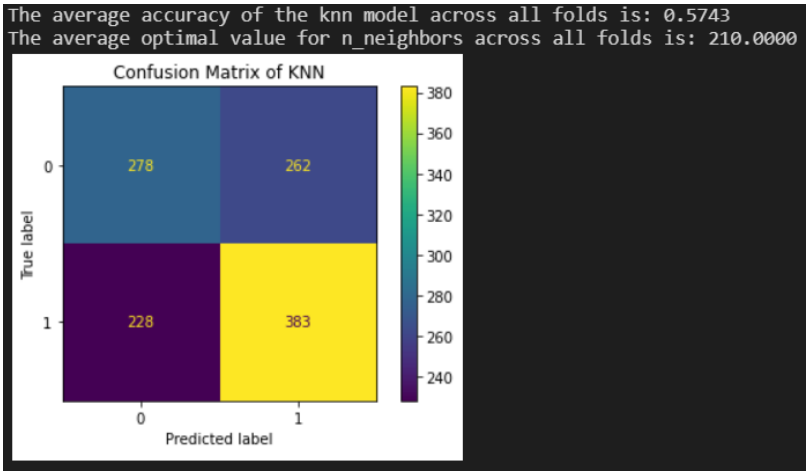


Figure 7: KNN results and CM.

Combined Models

Finally, the combined model was developed, which ended up containing the highest accuracy out of all models, with a pretty good sensitivity and specificity. One interesting thing about this model is that because only three individual estimators were used, it two of the estimators are poorly tuned, then the performance of the voting classifier will also be brought down because only one classifier works well.

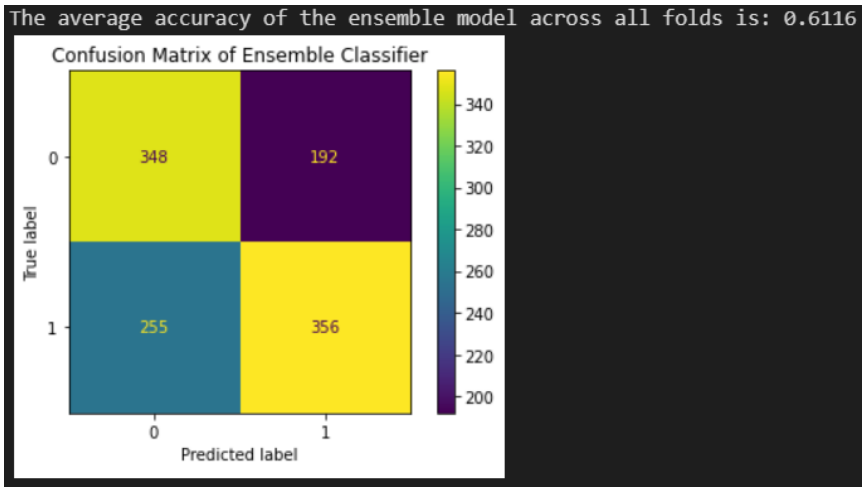


Figure 8: Voting classifier results and CM.

Conclusion

In conclusion, there is a lot of potential to developing a DR classifier using existing machine learning methods and models. With the results obtained earlier, there is a best case scenario of a 61.16% accuracy using a combined classification model. The accuracy, sensitivity, and specificity should all be higher for such a classifier to be of actual relevance, but this project showcases how powerful even a low-feature dataset can be.

Something that might be more useful is to include even more physiological features in DR related datasets, such as the age of the patient, sex, number of years diagnosed with diabetes, or even something like number of health checkups per year. This would not only make the classifier a more powerful tool, but also provide greater insight into what can be an early predictor of DR, and provide that information to help those who may have diabetes but do not suffer from DR.

References

[1] Kierstan Boyd: What Is Diabetic Retinopathy? Sep. 2020, https://www.aao.org/eye-health/diseases/what-is-diabetic-retinopathy

[2] Balint Antal, Andras Hajdu: An ensemble-based system for automatic screening of diabetic retinopathy, Knowledge-Based Systems 60 (April 2014), 20-27.

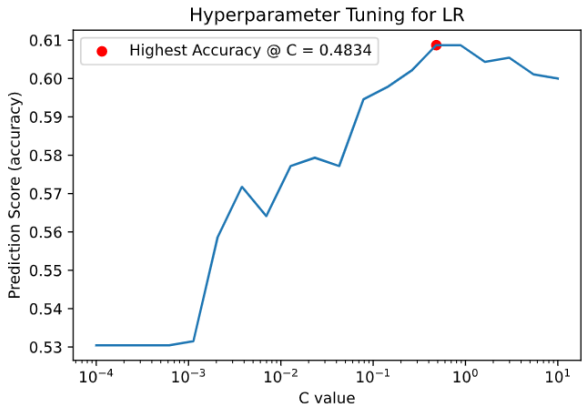
[3] "Sklearn.linear_model.LogisticRegression." *Scikit-Learn*, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression

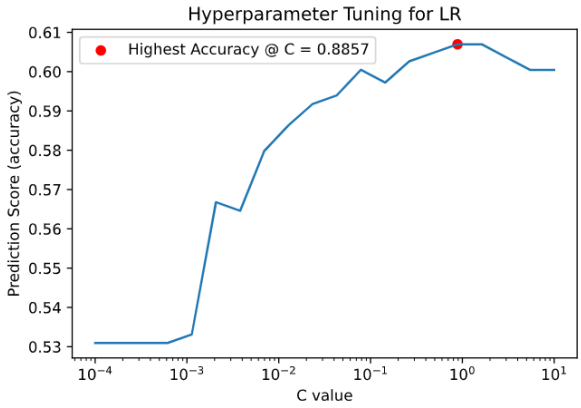
[4] "Sklearn.svm.SVC." *Scikit-Learn*, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC

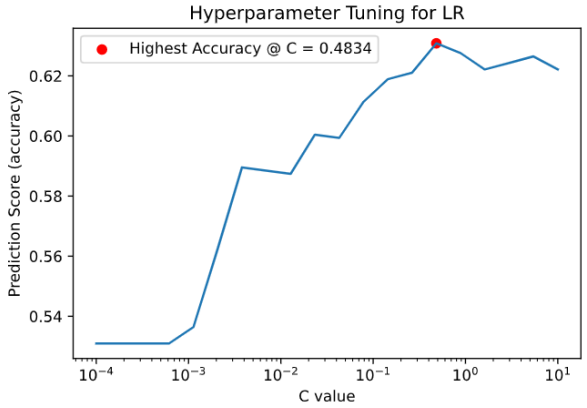
[5] "Sklearn.neighbors.KNeighborsClassifier." *Scikit-Learn*, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

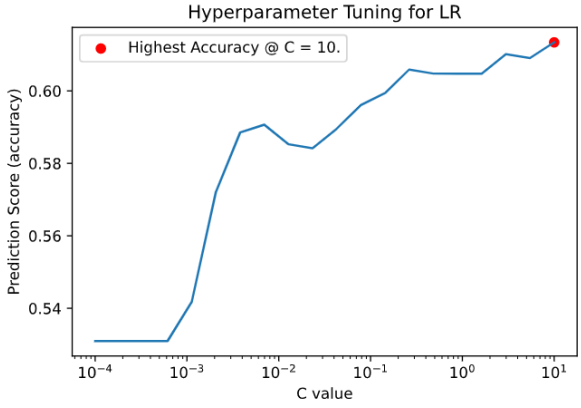
Appendix

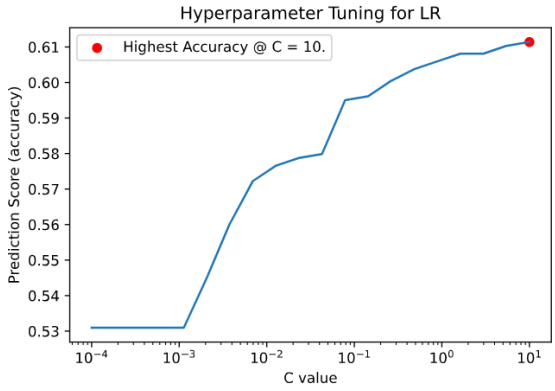
Logistic Regression Tuned Results:



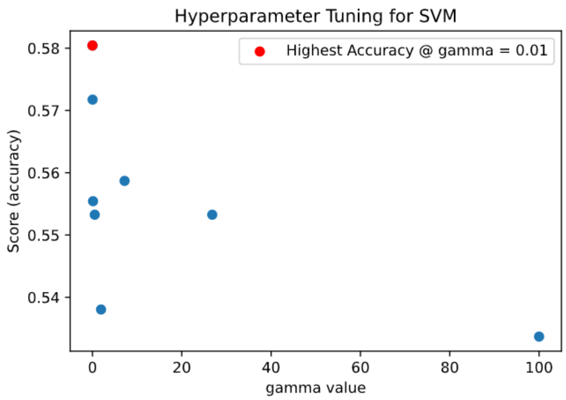


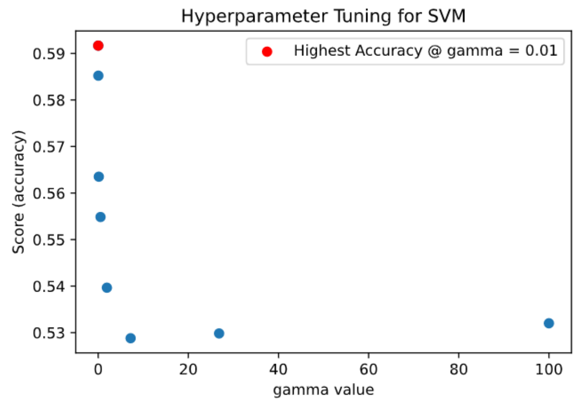


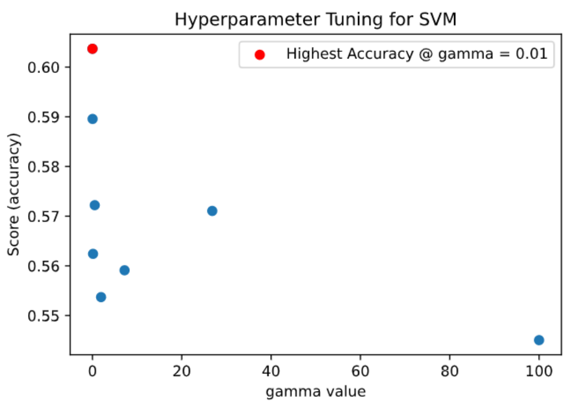


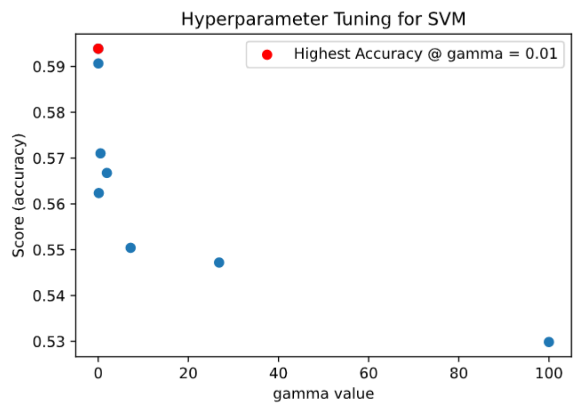


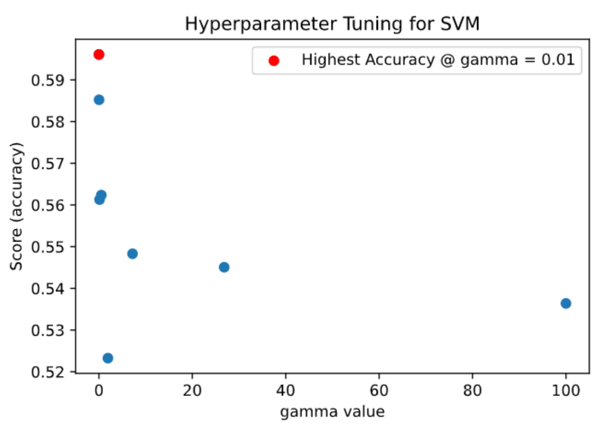
Support Vector Machine Tuned Results:











KNN Classifier Tuned Results: