Optimization for Machine Learning
Final Project
Instructor: Mert Gurbuzbalaban
Fall 2020
December 16, 2020

Mihika Gupta & Vidita Gawade

OBJECTIVE

“Predicting COVID Mortality with Machine Learning using optimization techniques”
The goal is to predict patient mortality based on COVID dataset available through kaggle. This project is inspired by a research paper recently published: “Predicting CoVID-19 community mortality risk using machine learning and development of an online prognostic tool, 2020 by Das et al.

1. Data preparation & cleaning

a. DATA Version 1 :Mihika:

After going through the Data available through KAGGLE, we decided to move ahead with the PatientInfo. Csv file, as it contained all the relevant attributes needed for our analysis. As discussed in class, we tried to follow the guidelines from the paper “Predicting CoVID-19 community mortality risk using machine learning and development of an online prognostic tool”, and our Final dataset had the following predictor attributes corresponding to it:

1. Patient Age
2. Patient Gender
3. Patient’s Province
4. Source of exposure to COVID Virus
5. Mortality: Binary categories(Patient Deceased=1, Patient Survived=0)

Since all the attributes had categorical values, we used various python functions to convert them to numeric factors for fitting into our models.Further a binary attribute mortality was created having values 0 and 1 using the “state” column from the original dataset which gave info about whether the patient was deceased, released or isolated.
Further all NaN’s were removed from this dataset, but no limit was put on the onset date of the disease of the patient is this version of the dataset. The dataset was found to be imbalanced and stratification and SMOTE techniques were applied to build Models. However all model evaluation metrics are compared later.

b. DATA Version 2 : Vidita:

After Mihika’s data cleaning, we saw that the paper filtered and used data for patients that had a symptom onset date through May 30, 2020 online so we removed observations that had any symptom onset after this date. We recognize that this can retain ‘NaN’ patients in this column which keeps us at a final matrix of 3736 rows by 4 input features and 1 response variable of mortality. Furthermore, we deleted any rows with ‘NaN’ based on only gender and age column as explained in paper. Filtered further on date on symptom onset as explained in paper

Next we recognized that the dataset is also highly imbalanced as there are very few deaths compared to surviving patients. The paper explained “SMOTE” analysis using the library called imblearn that creates synthetic data based on nearest neighbors of the underrepresented class. We implemented SMOTE into our dataset to have a better balanced representation of mortal and surviving patients. We recognize that this is helpful when the two classes have linear separability as the synthetic data that gets produced does so in a patterned manner. If the original dataset is coming from a random data set then it is not beneficial to use SMOTE.

2. Model building: Logistic Regression, SVM with Kernels, Gradient Boosting (Decision Tree based)

Before we train the models, we split up the SMOTE generated dataset into training and test set by sending 80% of the dataset to train and keeping 20% for test set along with using stratified sampling to ensure that there’s the same ratio of positive and negative classes in both the train and test sets.

The Model Evaluation Metrics that were produced are as follows:

Explanation of metrics:

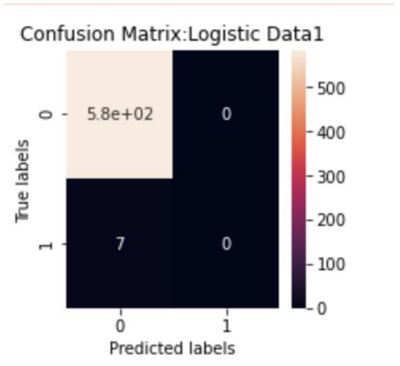
Accuracy: Overall accuracy is how well the classifier predicted correct classes out of total number in the dataset.
AUC: The area under the receiver operating characteristics curves measures how well the classifier distinguishes among the classes by plotting true positive rate against false positive rate. The closer the AUC is to 1 the better the classifier performed.
Calibration based on Matthews Correlation coefficient: To measure calibration, we used matthews correlation coefficient to measure quality of binary classifications. Typical logistic regression calibrates well as it optimizes log-loss to give balanced probabilities between 0 and 1, compared to Gaussian naive bayes which tends to push probabilities towards 0 or 1. Ideally we want a model that does well on both discrimination (AUC) and calibrartion (Matthews Correlation coefficient metric).
Specificity: This measures how accurate people that survived were truly predicted to survive.
Sensitivity: This measures how accurate people that died were truly predicted as dead.

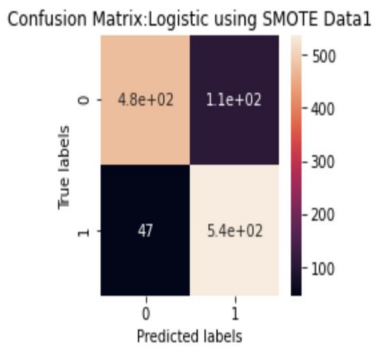
1. Logistic regression

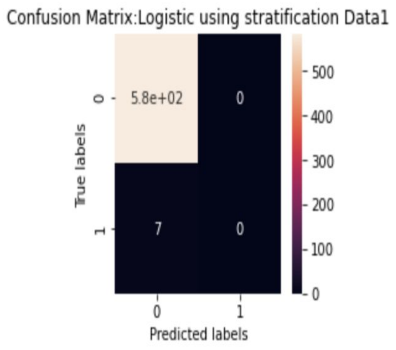
It is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

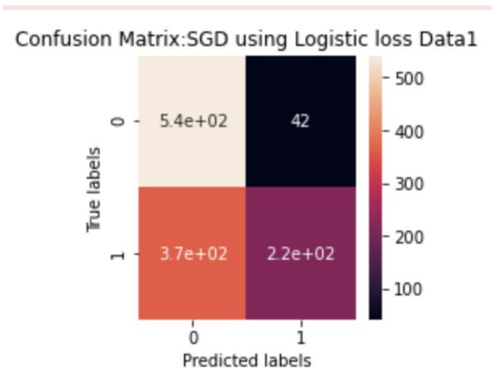
RESULTS

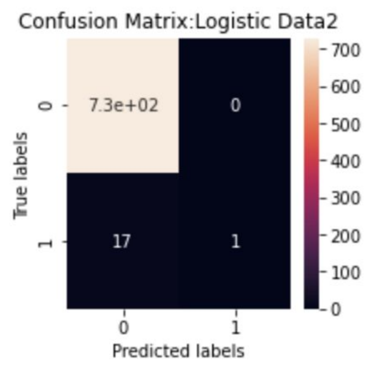
1. **Confusion Matrix Comparisons for all Models**

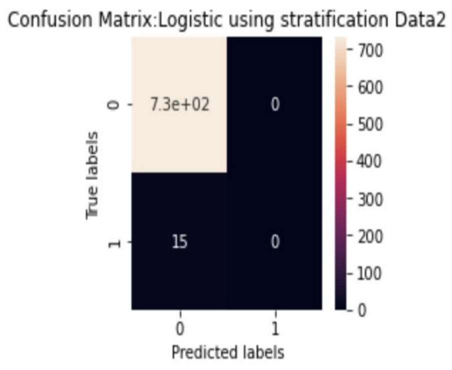


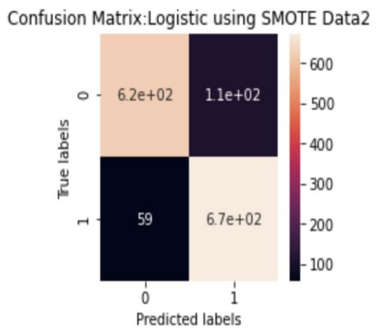


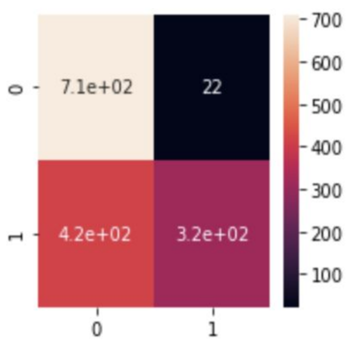




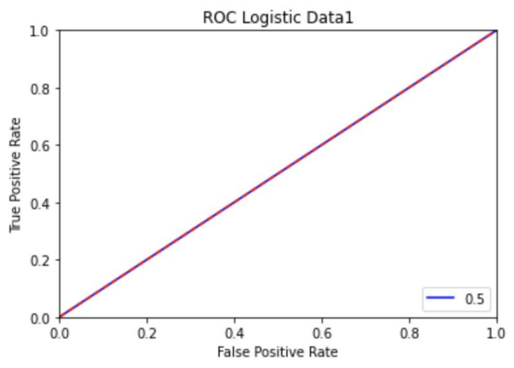


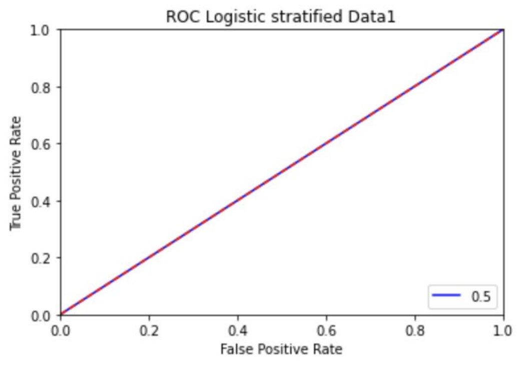


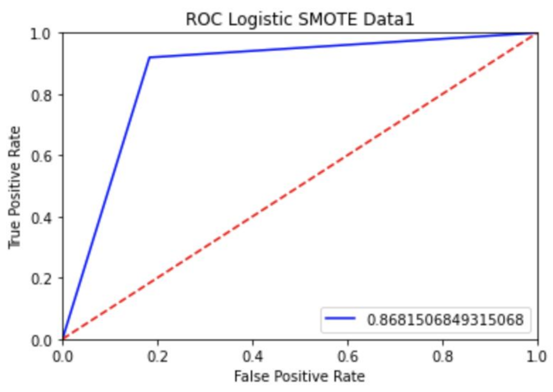


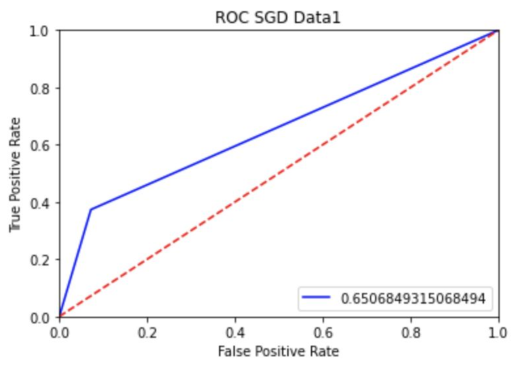


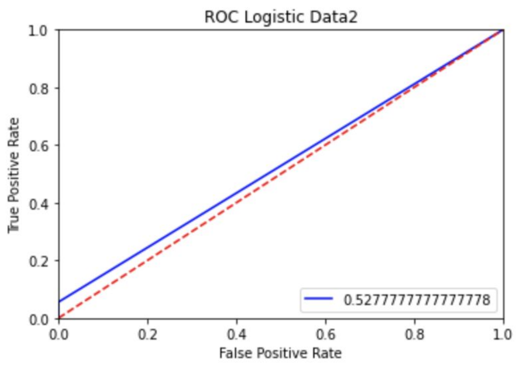
2. ROC Curve and Area Under Curve Comparisons

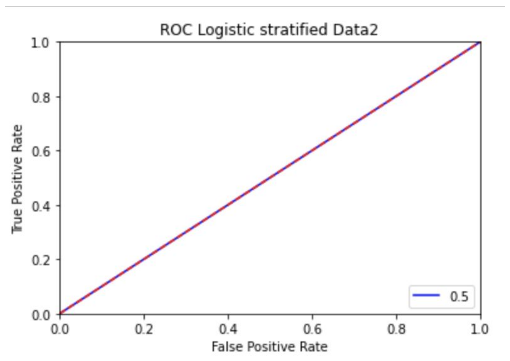


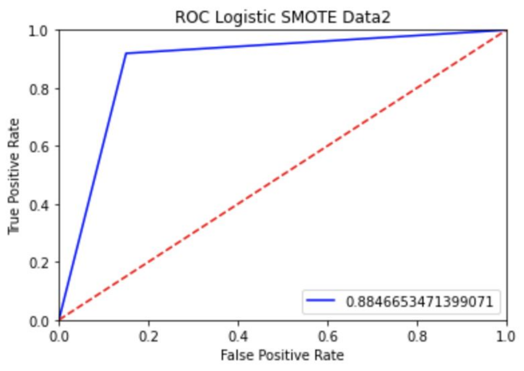


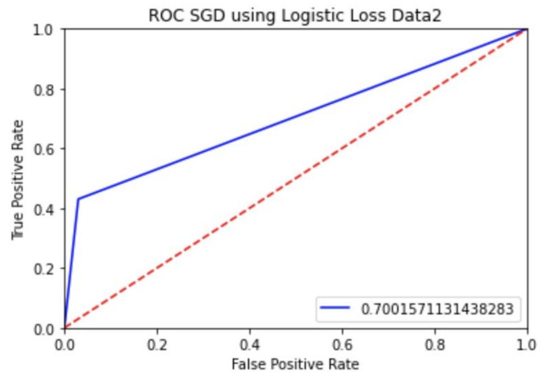




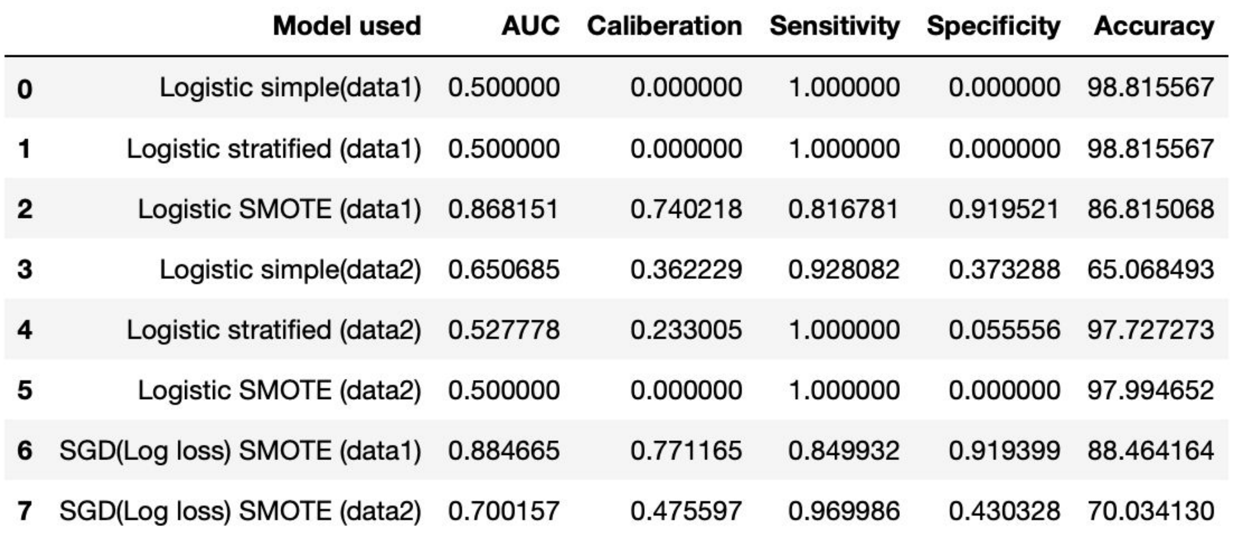








3. **Final Evaluation Metric Comparisons**



	Model used	AUC	Caliberation	Sensitivity	Specificity	Accuracy
0	Logistic simple(data1)	0.500000	0.000000	1.000000	0.000000	98.815567
1	Logistic stratified (data1)	0.500000	0.000000	1.000000	0.000000	98.815567
2	Logistic SMOTE (data1)	0.868151	0.740218	0.816781	0.919521	86.815068
3	Logistic simple(data2)	0.650685	0.362229	0.928082	0.373288	65.068493
4	Logistic stratified (data2)	0.527778	0.233005	1.000000	0.055556	97.727273
5	Logistic SMOTE (data2)	0.500000	0.000000	1.000000	0.000000	97.994652
6	SGD(Log loss) SMOTE (data1)	0.884665	0.771165	0.849932	0.919399	88.464164
7	SGD(Log loss) SMOTE (data2)	0.700157	0.475597	0.969986	0.430328	70.034130

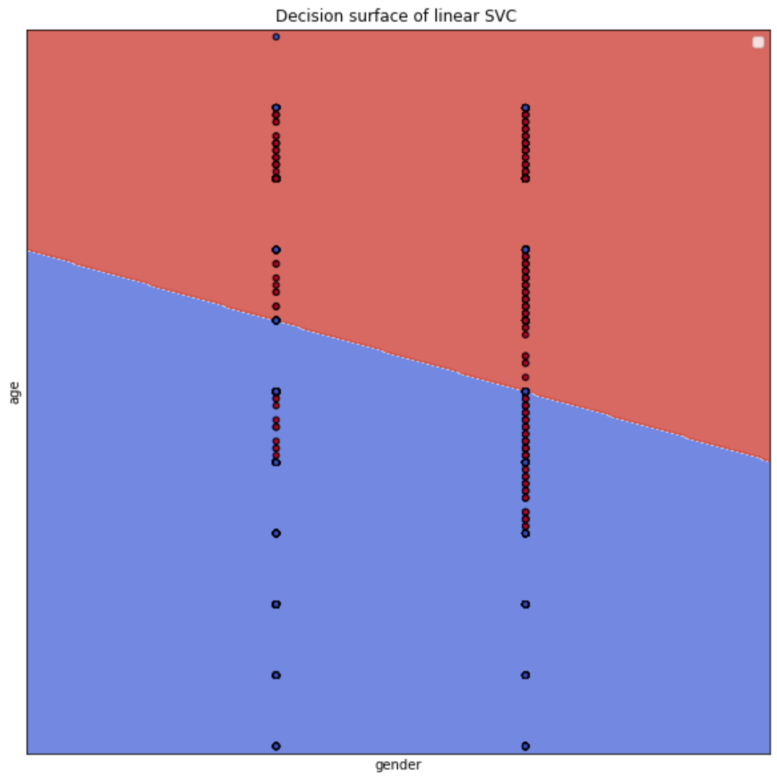
According to the above Comparison Table, the best Model was found to be Model 6, Stochastic Gradient Descent using Logistic Loss(SMOTE) over data1, followed by Model 2, Logistic Regression(SMOTE) over data1

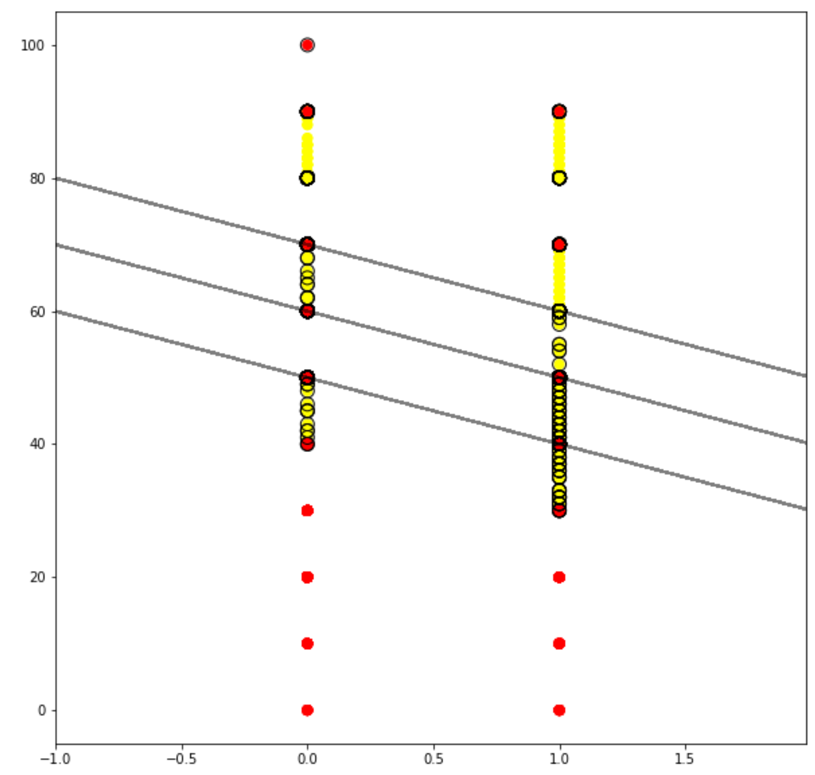
2. Support Vector Machines

In support vector machines, a hyperplane is built that classifies the response variable using duality linear programming as its basic foundation. Using different kernels, the original input space moves from one space to a transformed feature space. In our case, we tried out the linear kernel space, polynomial kernel space, and RBF kernel space. Apart from specifying the kernels, we kept default parameters.

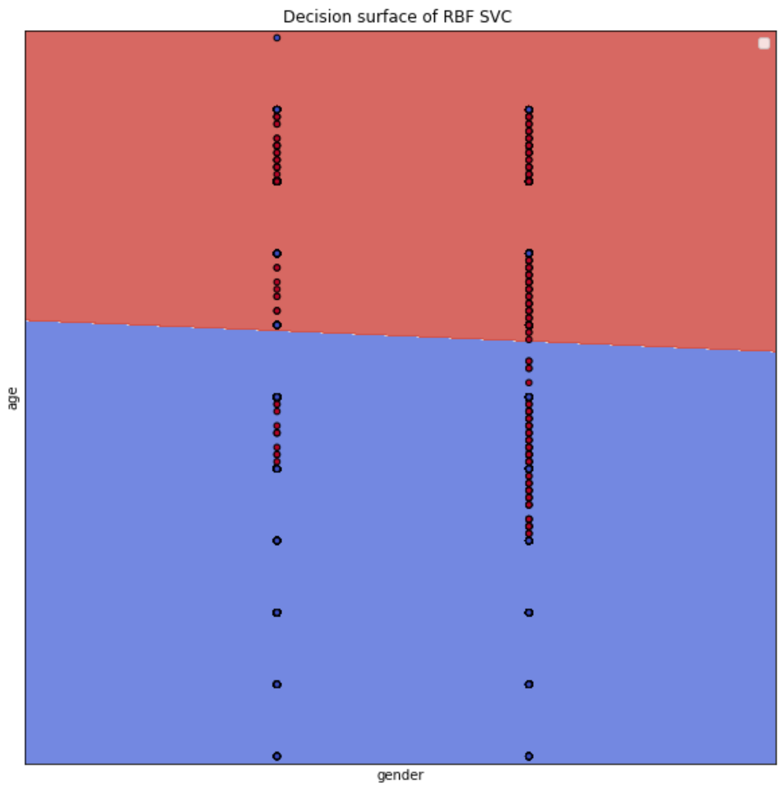
Since we have four input features and it is difficult to visualize classifier in more than three dimensions, we visualize a classifier on just two features: gender and age.

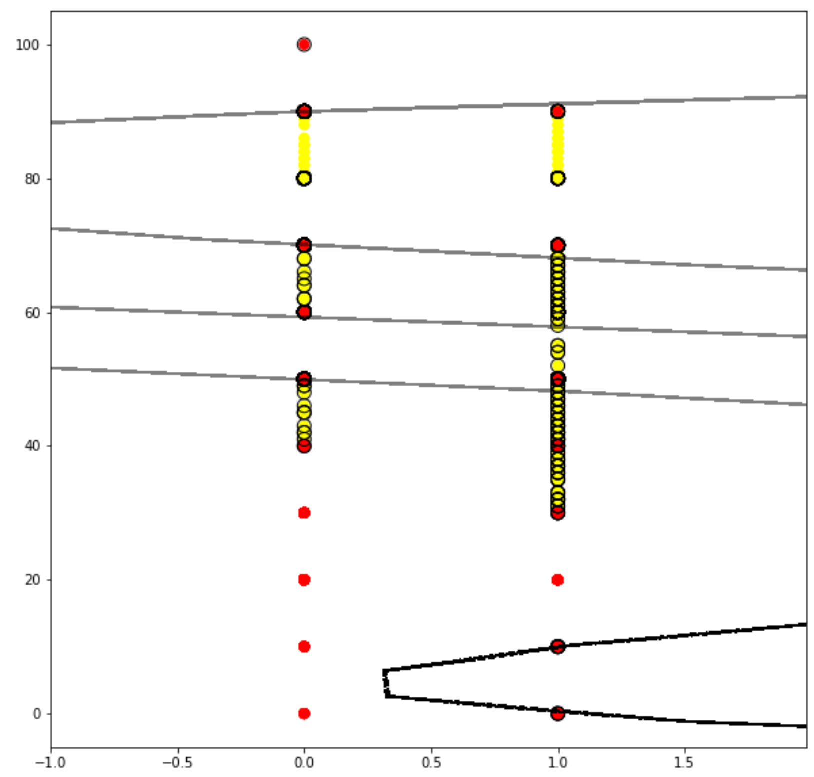
Linear kernel SVM:



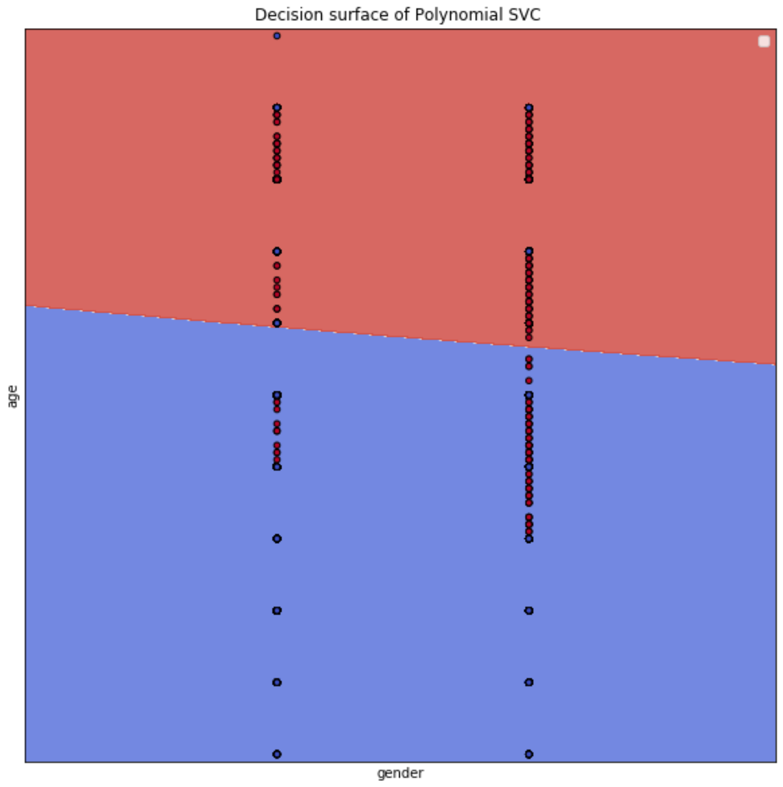


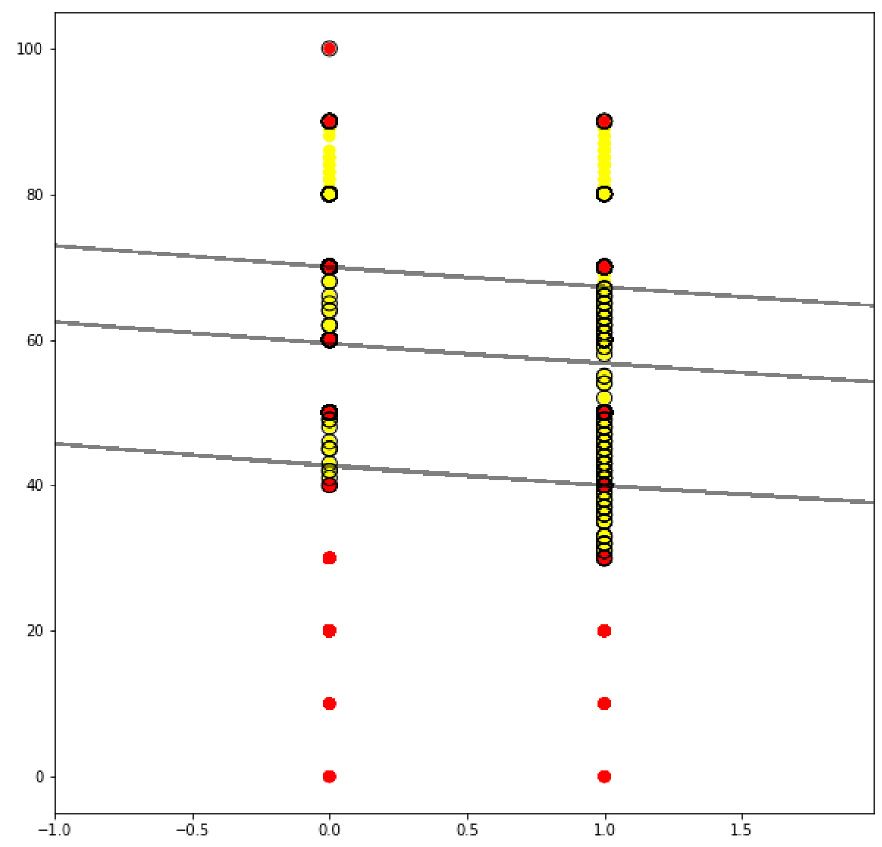
RBF Kernel SVM:



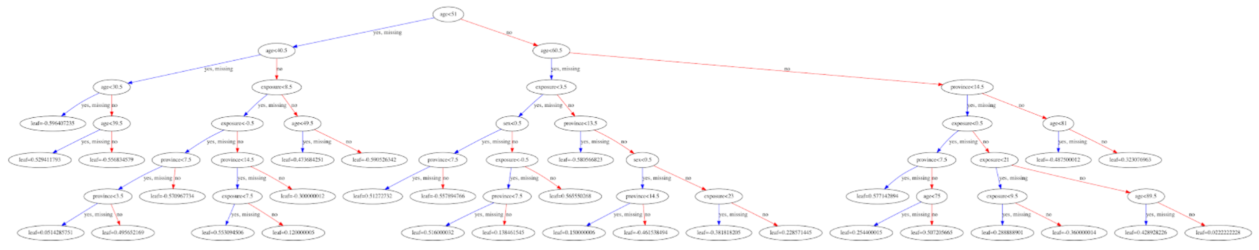


Polynomial Kernel SVM:





We tried gradient boosting as a last model that is a decision tree based model that uses a combination of decision trees by generating a recursive partition of predictors. Each decision tree learns from its previous tree model and passes an improved objective function to the new boosted classifier. We kept the default parameters in this case as well. An example of a gradient boosted tree classifier is shown as follows:



Final table:

Model type & features used & parameters	Train accuracy	Train AUC	Train calibration	Train specificity	Train sensitivity	Test “”	Test “”	Test “”	Test “”	Test “”

SVM Linear, all features, default parameters	87.86%	92%	76%	84%	91%	86%	92%	72%	82%	89%
SVM Polynomial, all features, default parameters	86.7%	92.5%	73%	86%	87%	85%	92.6%	71%	85%	86%
SVM RBF, all features, default parameters	87.76%	92%	76%	80%	95%	86.14%	92%	73%	78%	93%
Gradient Boosting, all features, default parameters	96%	99%	92%	93%	99%	95%	98%	91%	92%	98%

Analysis: When comparing the classifiers based on SVM and gradient boosting, gradient boosting outperformed the other classifiers based on given metrics as shown in table.

References:

Dataset: https://www.kaggle.com/kimjihoo/coronavirusdataset

Paper: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7528809/pdf/peerj-08-10083.pdf

Code for SMOTE:
https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Other machine learning libraries came from importing libraries available in Python sk-learn