Milestone 3: Report

Team 7
Logan Courtney, Terryl Dodson, Griffin Lumb, Ben Miller

Method 1: Logistic Regression

Features:

FEATURE	FEATURE DESCRIPTION
cdc_report_dt	This feature is the date that the cdc reported the instance's coronavirus test result.
pos_spec_dt	The pos_spec_dt feature is the date when the coronavirus test was conducted.
onset_dt	This feature is the date in which the individual started feeling symptoms.
current_status	The current status feature provides the instance's coronavirus test result.
sex	Sex is a binary classification feature describing if the individual is female or male.
age_group	This feature groups the instances into age groups by decade, ie. 0-9, 10-19, 20-29, etc.
Race and ethnicity	This feature is the instance's race and ethnicity.
hos_yn	Hos_yn is a binary classification of whether an instance was admitted to a hospital or not.
Icu_yn	Icu_yn is a binary classification feature which describes if an individual was admitted to the ICU or not.
death_yn	This feature is a binary classification of whether the individual perished or not. This will be our target feature.
medcond_yn	Medcond_yn is a binary classification feature which states if an individual had previous underlying medical conditions that contributed to the severity of the virus.

Equation

When we first started writing the script for the first method our initial method was linear regression. We had implemented linear regression in its entirety until we realized that it's incapable of classification. After consulting the lecture slides, we decided to convert it from Linear Regression to Logistic Regression. Logistic Regression works well with classification and calculating metrics such as true positives and negative and false positives and negatives. Below you will find the logistic function that was found in the slides. We didn't hard code the formula below, we decided to just import LogisticRegression and implement it that way.

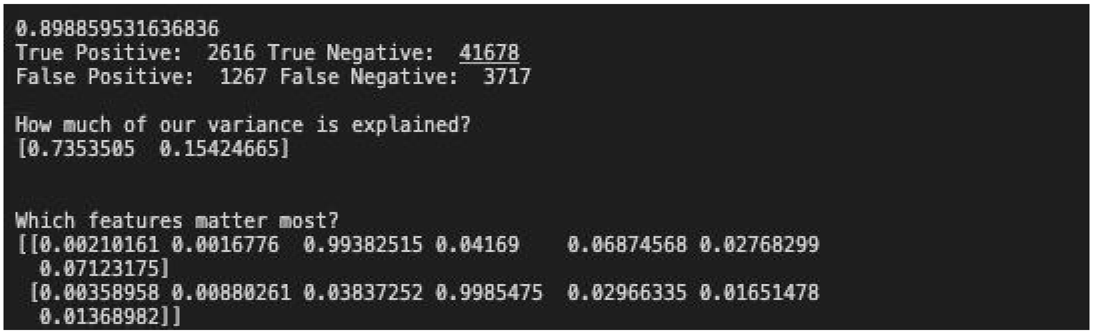
$$\sigma(z) = \frac{1}{1 + \exp[-z]} = \frac{\exp z}{1 + \exp z}$$

Pre-Processing & Implementation

We extracted 7 features and one target variable from our dataset using the library Pandas. Each of the features had string labels like "Yes" or "No" and other ethnicity and age labels, so we had to convert those into numerical values. After standardizing and formatting our data correctly, we separated it with 80% of the data being used for training and 20% being used to test the model. Then, we proceeded to train a Logistic Regression model from Sklearn. After the model is trained, we calculate various scores like accuracy, true positives, true negatives, false positives, and false negatives.

Evaluation

As stated above, Logistic Regression works well with classification and metrics which is why we decided to go this route. When you run the script, it will print out four separate sections. The first section is what the model predicted for the 20% used for testing (49,278 instances). After it predicts the 49,278 instances, it then prints out the accuracy and the metrics. As you can see the model performed well. The accuracy turned out to be 89.8% and we falsely predicted 4,984 instances in total out of 49,278 which isn't that bad at all. We also decided to implement PCA just out of curiosity. After implementing PCA, we discovered that after diminishing our data to two components we still have 88% of the data variance explained which means we only lost 12%. We also discovered that in the first component age_group was the most significant, and for the second component race_ethnicity was the most significant. This had us confused for a while because prior to converting over to Logistic Regression, Linear Regression had classified the icu feature to be the most important which makes sense. Our other two group members also said that after implementing the decision tree, they received icu as the most significant feature as well. However, we decided to stick with the PCA and explain how icu makes the most sense over age_group and race_ethnicity when it comes to most significant features. Below, you will find a screenshot of the data that was discussed above. If you would like to view the predicted instances, you can run the LogisticRegressionModel script and the predicted instances will be printed along with the metrics that are in the screenshot below.

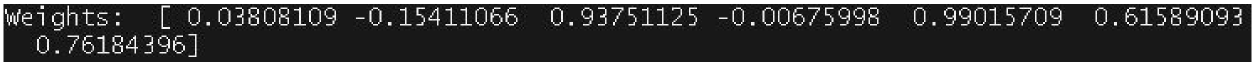


```
0.898859531636836
True Positive:  2616 True Negative:  41678
False Positive:  1267 False Negative:  3717

How much of our variance is explained?
[0.7353505  0.15424665]


Which features matter most?
[[0.00210161 0.0016776  0.99382515 0.04169    0.06874568 0.02768299
  0.07123175]
 [0.00358958 0.00880261 0.03837252 0.9985475  0.02966335 0.01651478
  0.01368982]]
```

While PCA provided us with confusing results, we were able to print out the weights of our logistic regression model. The weights are in order accordingly: current status, sex, age group, race and ethnicity, hospitalization, ICU, and medical condition.



```
Weights:  [ 0.03808109 -0.15411066  0.93751125 -0.00675998  0.99015709  0.61589093
  0.76184396]
```

We can see that the age group and whether or not the patient was hospitalized are weighted the heaviest in our model. This makes a lot more sense than the results of the PCA. The results of implementing the logistic regression model seem really promising. We would have never expected to be able to get near 90% accuracy on such a practical dataset. We have expanded our knowledge on regression-based algorithms along with other ML methods through this project.

Method 2: Decision Trees

Equation

When considering the decision tree algorithm, we are posed with the choice between using gini impurity or entropy. With an already taxed computational cost as a result of the large data set, we decided to use gini impurity to save time since we would not need to compute logarithmic functions. We use gini impurity to predict the probability of misclassifying on a split. The gini impurity equations is as follows:

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} \left(p_i \sum_{k \neq i} p_k \right) = \sum_{i=1}^{J} p_i (1 - p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

Pre-processing Steps

When constructing our classification algorithm we considered multiple avenues. At first, we had trouble implementing the decision tree with the given features, however, we ran into issues involving the nominal multivariate categorical features. As a result, we used a Pandas feature called “get_dummies” which takes multivariate categorical features and splits them into binary features. Additionally, we removed the “cdc_report_dt”, “pos_spec_dt”, and “onset_dt” features as they offered little to no information gain. As a result of the “get_dummies” function, we trained our decision tree classifier (sklearn) on 20 features. Moreover, we split our data into 90% training and 10% test data, with the training set containing 221,781 samples and the test set containing 24,643 samples. After training the decision tree classifier, we predict using the test data and calculate several metrics, such as accuracy, false negatives and positives, and true negatives and positives.

Implementation

Our dataset contains primarily categorical data. Therefore, we decided to implement a decision tree model as it is a classification algorithm that can handle categorical data. The main workload for implementing the decision tree classifier was working with the temperamental dataset. We first read in our dataset, dropping the first three columns (as listed in pre-processing). Following, we divide our dataset into x and y training and test sets, respectively. Next, we iterative over each dataset and replace the categorical features with their respective binary counterpart. We use the Pandas get_dummies function to convert the categorical variables into dummy/indicator variables for the decision tree classifier. After this encoding, our datasets reached 20 features. Following, we train the decision tree on the training dataset. When training the decision tree, we selected a max tree depth of 4. We chose this tree depth to both reduce the algorithm’s runtime and also because this tree depth maximized our accuracy score in the prediction stage. We used Graphviz in association with sklearn to get a visual representation of our decision tree:

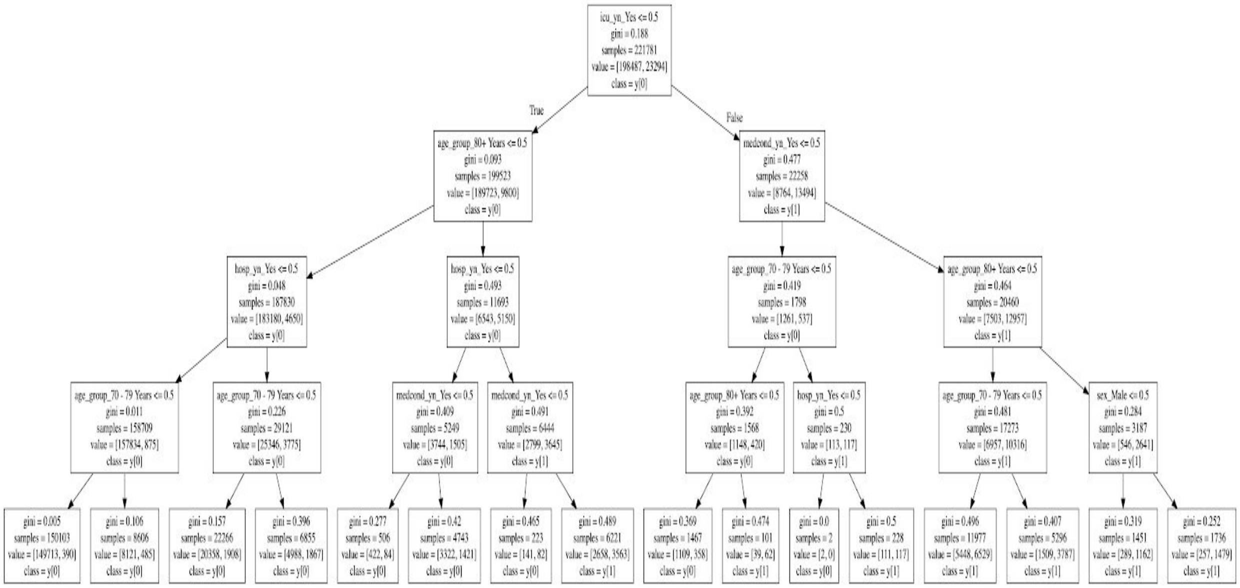

icu_yn_Yes <= 0.5 gini = 0.188 samples = 221781 value = [198487, 23294] class = y[0]
True
False
age_group_80+ Years <= 0.5 gini = 0.093 samples = 199523 value = [189723, 9800] class = y[0]
medcond_yn_Yes <= 0.5 gini = 0.477 samples = 22258 value = [8764, 13494] class = y[1]
hosp_yn_Yes <= 0.5 gini = 0.048 samples = 187830 value = [183180, 4650] class = y[0]
hosp_yn_Yes <= 0.5 gini = 0.493 samples = 11693 value = [6543, 5150] class = y[0]
age_group_70 - 79 Years <= 0.5 gini = 0.419 samples = 1798 value = [1261, 537] class = y[0]
age_group_80+ Years <= 0.5 gini = 0.464 samples = 20460 value = [7503, 12957] class = y[1]
age_group_70 - 79 Years <= 0.5 gini = 0.011 samples = 158709 value = [157834, 875] class = y[0]
age_group_70 - 79 Years <= 0.5 gini = 0.226 samples = 29121 value = [25346, 3775] class = y[0]
medcond_yn_Yes <= 0.5 gini = 0.409 samples = 5249 value = [3744, 1505] class = y[0]
medcond_yn_Yes <= 0.5 gini = 0.491 samples = 6444 value = [2799, 3645] class = y[1]
age_group_80+ Years <= 0.5 gini = 0.392 samples = 1568 value = [1148, 420] class = y[0]
hosp_yn_Yes <= 0.5 gini = 0.5 samples = 230 value = [113, 117] class = y[1]
age_group_70 - 79 Years <= 0.5 gini = 0.481 samples = 17273 value = [6957, 10316] class = y[1]
sex_Male <= 0.5 gini = 0.284 samples = 3187 value = [546, 2641] class = y[1]
gini = 0.005 samples = 150103 value = [149713, 390] class = y[0]
gini = 0.106 samples = 8606 value = [8121, 485] class = y[0]
gini = 0.157 samples = 22266 value = [20358, 1908] class = y[0]
gini = 0.396 samples = 6855 value = [4988, 1867] class = y[0]
gini = 0.277 samples = 506 value = [422, 84] class = y[0]
gini = 0.42 samples = 4743 value = [3322, 1421] class = y[0]
gini = 0.465 samples = 223 value = [141, 82] class = y[0]
gini = 0.489 samples = 6221 value = [2658, 3563] class = y[1]
gini = 0.369 samples = 1467 value = [1109, 358] class = y[0]
gini = 0.474 samples = 101 value = [39, 62] class = y[1]
gini = 0.0 samples = 2 value = [2, 0] class = y[0]
gini = 0.5 samples = 228 value = [111, 117] class = y[1]
gini = 0.496 samples = 11977 value = [5448, 6529] class = y[1]
gini = 0.407 samples = 5296 value = [1509, 3787] class = y[1]
gini = 0.319 samples = 1451 value = [289, 1162] class = y[1]
gini = 0.252 samples = 1736 value = [257, 1479] class = y[1]

Image link for ease of viewing: https://ibb.co/9q5fNRp

Evaluation

When running our decision tree algorithm, a DOT file is created with the above decision tree model. This model shows each node’s split feature, gini impurity, and remaining sample size of the data. After iterating through several different max tree depths, we set the max depth threshold to 4. We do this because our model with a max tree depth of 4 both reduced computation costs and returned the highest accuracy score when predicting the test dataset. Our algorithm also computes and outputs the number of true and false positives and negatives. We output these values using a confusion matrix. Using our decision tree, we found our accuracy to be:

```
Accuracy = 0.9265917299030151
```

Furthermore, the confusion matrix is as follows:

CONFUSION MATRIX	Predicted Dead	Predicted Survival
True Dead	21725	1394
True Survival	415	1109

With an accuracy of 92.6%, we are satisfied with how our algorithm performed. Out of the test set with 24,643 patients, we accurately predicted 22,834 cases. We find the biggest indicator for COVID susceptibility is the icu feature, which correlates with our team's logistic regression model. The next most significant indicators for COVID susceptibility are age_group and prior med_cond. Specifically, the age group of 80+ years is extremely susceptible to COVID. In addition, the feature hosp_yn was also a significant splitting criterion. As most of these splitting criteria correlate with the logistic regression model, we find our decision tree model to be fairly accurate. Considering our practical dataset, we find this model to be promising for predicting susceptibility to COVID.

Conclusions

In conclusion, our goal was to create two machine learning algorithms that will be able to predict the susceptible of a patient contracting COVID due to the following features: current status, sex, age group, race and ethnicity, hospitalization, ICU, and medical condition. In order to accomplish this goal, we decided to implement a Logistic Regression model and a decision tree classifier. Our group is very pleased with how well our models turned out. Given the dataset and the features mentioned above, we were able to predict if patients were susceptible to COVID with a 90% accuracy with the logistic regression model and 93% accuracy with the decision tree classifier. Given the high accuracies for both of these models, we believe that they are both effective implementations to predict susceptibility to COVID.

Team Member Contributions

Terryl Dodson and Logan Courtney - Logistic Regression Script & Method 1 of Report
Ben Miller and Griffin Lumb - Decision Trees Script & Method 2 of Report