COVID-19: Classifying Risk Level

Tucker R. Surdock

Washington State University

CPT_S 315 Introduction to Data Mining

Instructor*: Reet Barik*

December 2022

Introduction

According to the CDC website, “COVID-19 (coronavirus disease 2019) is a disease caused by a virus named SARS-CoV-2 and was discovered in December 2019 in Wuhan, China. It is very contagious and has quickly spread around the world.”[1] This disease may only cause minor symptoms similar to the flu or a cold. However, for some people, it can lead to severe illness and death. So, the motivation for this project is to analyze a data set of COVID patients, to assess a risk level. This can help doctors decide who needs admission into the hospital since there are limited resources in some cases. In the past few years during the pandemic, many people believed COVID to be a ‘hoax’ and refused to take precautions for public health. My personal motivation is my belief that humans should care about public health and take it seriously. My goal is to create a machine learning model that can learn from COVID data to classify people's risk level.

Data Mining Task

I found a data set on kaggle, an open source data set website for developers, that was provided by the Mexican government. The dataset consists of 21 unique features and 1,048,576 unique patients.[2] The data set also provides information about patients pre-existing conditions. The input data will be in the form of a comma separated file. Values of 1 and 2 are used as boolean

[1] Centers for Disease Control and Prevention. (n.d.). *Basics of covid-19*. Centers for Disease Control and Prevention. Retrieved December 10, 2022, from https://www.cdc.gov/coronavirus/2019-ncov/your-health/about-covid-19/basics-covid-19.html

[2] Nizri, M. (2022, November 13). *Covid-19 dataset*. Kaggle. Retrieved December 10, 2022, from https://www.kaggle.com/datasets/meirnizri/covid19-dataset

functions for whether a patient has the unique feature or not. Values for non-boolean features are in normal integer/string forms.

	A	B	C	D	E
1	USMER	MEDICAL_	SEX	PATIENT_	DATE_DIE
2	2	1	1	1	3/5/2020
3	2	1	2	1	3/6/2020
4	2	1	2	2	9/6/2020
5	2	1	1	1	#########
6	2	1	2	1	21/06/202

Figure 1: example of the raw input data set, imported to excel

As one can see, the data set needs cleaning up due to invalid values being used for boolean functions and dates of death.

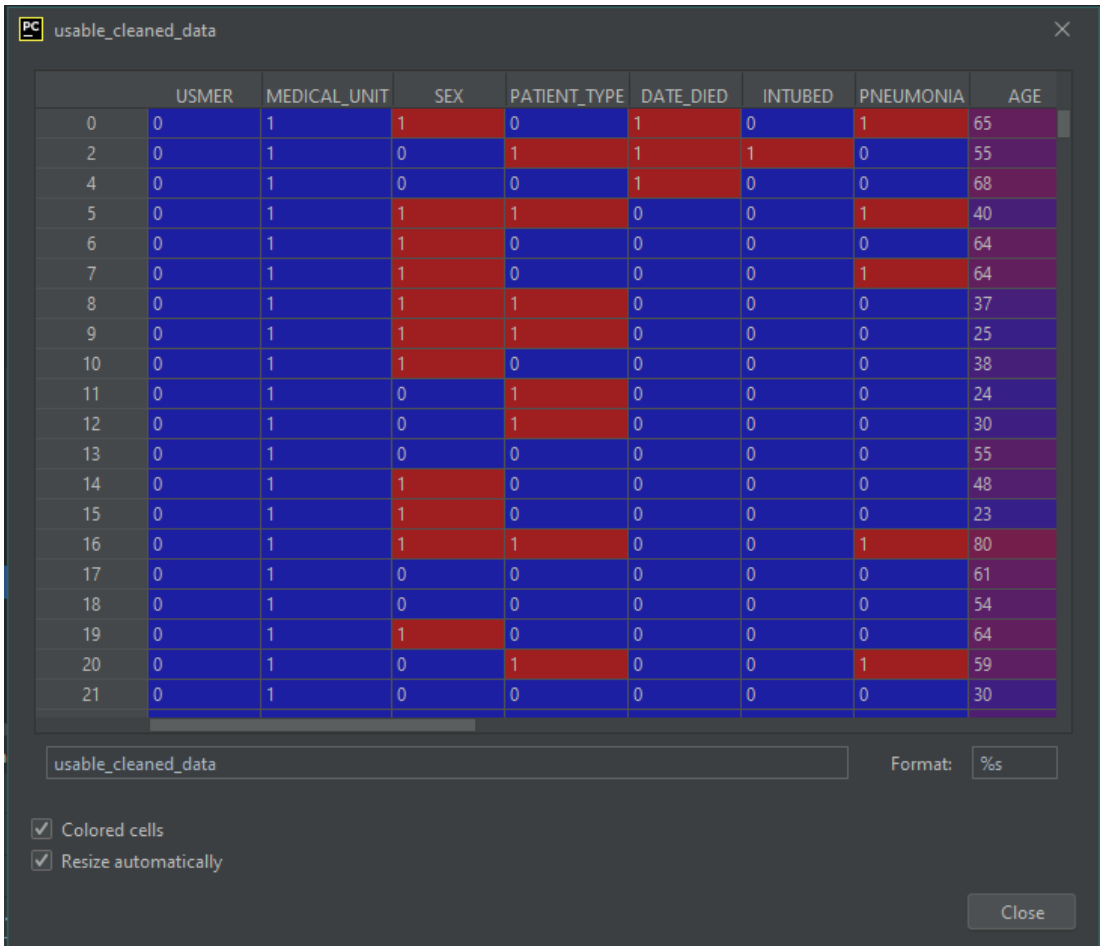


Figure 2: cleaned up data converted to binary values for boolean features (DataFrame view using PyCharm debugger)

The questions that I am going to investigate:

Which methods for learning are most effective with this data set?

What kernel setting works best for the SVM model? (Linear, poly, or rbf)

When using KNN algorithm, how do different K values affect the accuracy?

My machine learning system shall output a bar graph comparing the accuracy (f-score) of the three different models implemented. The system shall also output a bar graph comparing the SVM models with different kernels. Finally, it shall output a line graph of test accuracies for each different k-values used with the KNN algorithm.

The python framework Scikit does not have a built-in KNN model. So, my largest challenge was creating my own model. (Code is shown in Figure 3)

Technical Approach

As was explained in earlier readings for this course, I will be using multiple classification algorithms to get the best outputs. The first algorithm I will use is the Perceptron algorithm to determine if the patient is high risk or not, since it is a binary classifier. I will be using a multi class perceptron to analyze this complex data set that contains many features. The second algorithm that I will use is an SVM algorithm. I am choosing to use this algorithm as a classifier because SVM works well with many datasets. It also works well with high and low dimensional data. I will be using different kernels to decide the best one. The last algorithm I will use for

learning is the KNN (K-Nearest Neighbors) algorithm. I will use multiple distance algorithms to compare the Euclidean algorithm to.

The machine learning system that I am going to create, will use 90% of the cleaned up data to train and the other 10% to test. When training the models, high risk patients will be patients who have any or all three following features: The patient was admitted to the ICU, the patient was treated with a ventilator, or the patient must have passed away.

Pseudo code for my algorithms

All training for each algorithm will be utilizing undersampling suggested by kaggle user: Meir Nizri[3])

Also all the algorithms will utilize the Scikit[4] python framework and its documentation for ML.

Multi-class Perceptron algorithm:

1. Train the data set
2. Create a classifier and return the prediction
3. Return the f-score

SVM algorithm:

1. Train the data set
2. Initialize parameters for SVM
    a. Kernels is a list of kernel settings (linear, poly, rbf)

[3] https://www.kaggle.com/meirnizri
[4] https://scikit-learn.org/stable/modules/linear_model.html#perceptron

b. F -scores dictionary for all accuracies

c. Best result

3. For each kernel setting create an classifier and return the prediction
4. For each model calculate the f-score and return the best one

KNN algorithm:

Scikit does not provide a predefined KNN class. So I had to create my own model. The distance function that is used is the built in SciPy distance matrix algorithm that returns a matrix containing distances from every vector in x to every vector in y. By default the Minkowski p-norm used is p=2.

```
# class for creating a k-nearest neighbor model
class KNN(object):

    def __init__(self, k, x, y):
        self.k = k
        self.y_train = y
        self.x_train = x

    def predict(self, x):

        num_test = x.shape[0]
        num_train = self.x_train.shape[0]
        distances = np.zeros((num_test, num_train))
        distances = scipy.spatial.distance.euclidean(x, self.x_train)

        num_test = distances.shape[0]
        y_prediction = np.zeros(num_test)
        for i in range(num_test):
            closest_y = []
            k_nearest_ids = np.argsort(distances[i, :])[:self.k]
            closest_y = self.y_train[k_nearest_ids]
            y_prediction[i] = np.argmax(np.bincount(closest_y))
        return y_prediction

    def get_k_value(self):
        return self.k
```

Figure 3: screenshot of KNN model code

1. Train the data set

2. Initialize parameters
    a. K_vals is a list of different k values
    b. F scores
    c. Best score
3. For each k-value create a KNN model and return the prediction.
4. Compute the f-score for each model and return the best one

Evaluation Methodology

I found this data set on kaggle, an open source data set website for developers, that was provided by the Mexican government. “The dataset consists of 21 unique features and 1,048,576 unique patients.”[5] One challenge was that the data needed cleaning up, because there were a lot of invalid inputs (missing data) for some of the features. Also, the original person that posted this data set recommends undersampling when training. This is because the majority class is made up of patients that survived COVID after testing positive, this class makes up close to 93% of the data.

For each of the algorithms that I implemented in my project, they all received an f-score. The f-score is the accuracy of the model on the test data. Obviously, I chose to use f-scores because the purpose of this program in the real world is to predict someone’s risk level. When making a

[5] Nizri, M. (2022, November 13). *Covid-19 dataset*. Kaggle. Retrieved December 10, 2022, from https://www.kaggle.com/datasets/meirnizri/covid19-dataset

future prediction in the real world the model with the highest accuracy is what doctors would use to assess risk levels. Not only are these f-scores provided for each model for comparisons between each other, f-scores are provided for each variation of the algorithms. This allows for evaluation between the variations of each model, particularly the KNN and SVM models I implemented.

Results and Discussion

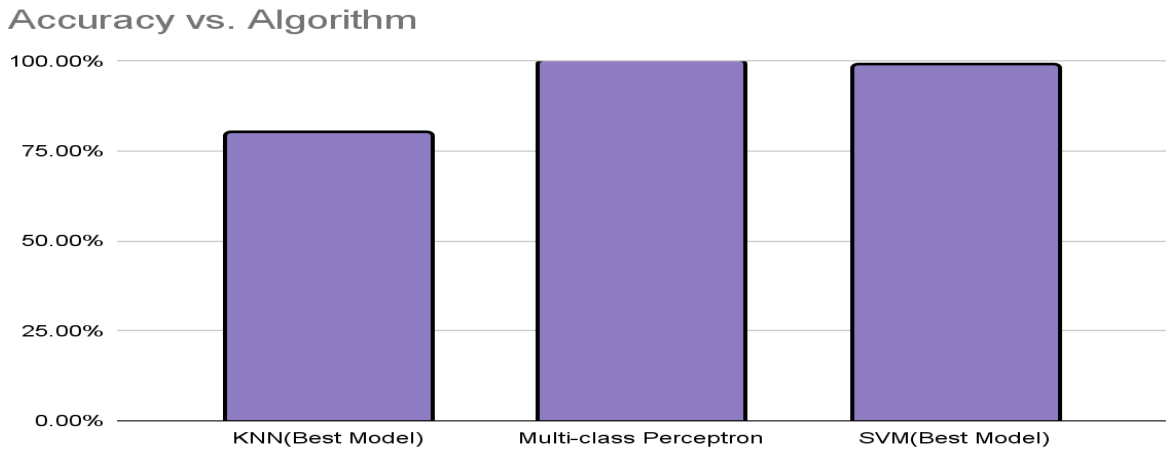
All of the algorithms printed the resulting accuracies labeled with their corresponding settings.

```
linear: Accuracy = 1.0
poly: Accuracy = 0.8209415584415585
rbf: Accuracy = 0.7429623065912344
```

Figure 4: example of output from SVM algorithm

Graphical representation of the results:

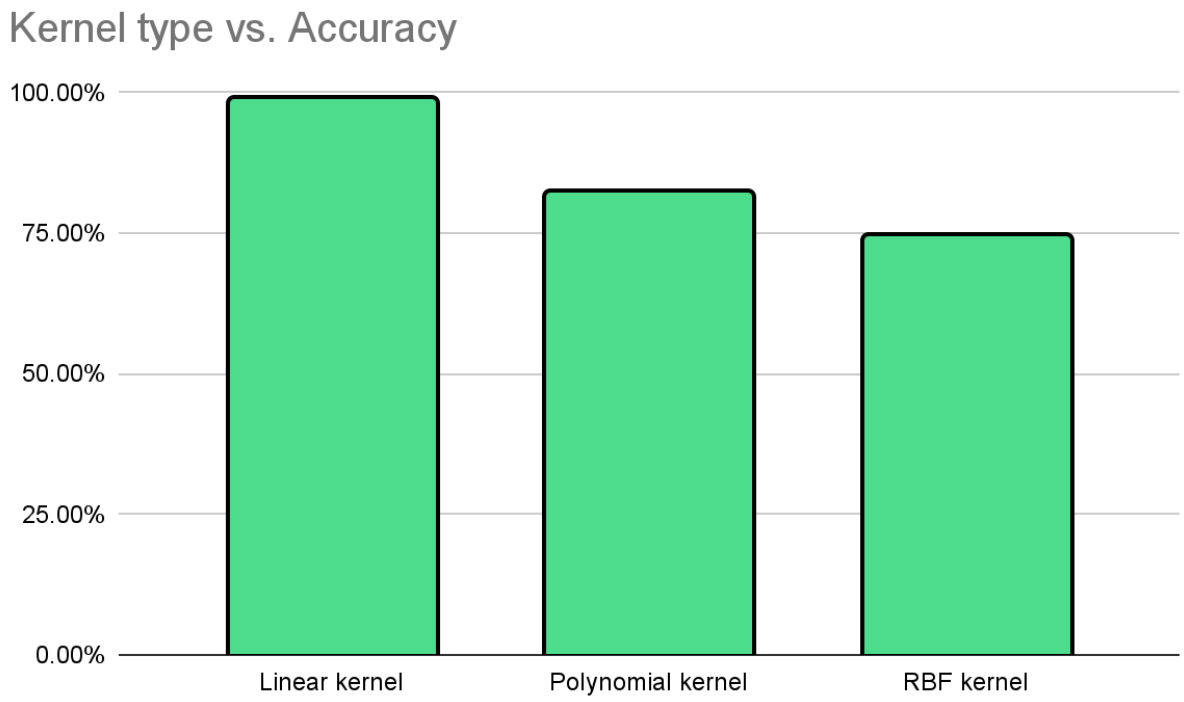
Which methods for learning are most effective with this data set?



Graph 1: comparison of best f-scores for each model

When comparing the best models of each algorithm I implemented, Multi-class perceptron performed the best with close to 100% accuracy. However, the worst performing algorithm used the KNN model that I created myself. This is probably the lowest, with such a high performance gap, because it was not a built-in model provided by SciKit. SVM also performed well with this data set, but only when the linear kernel was used.

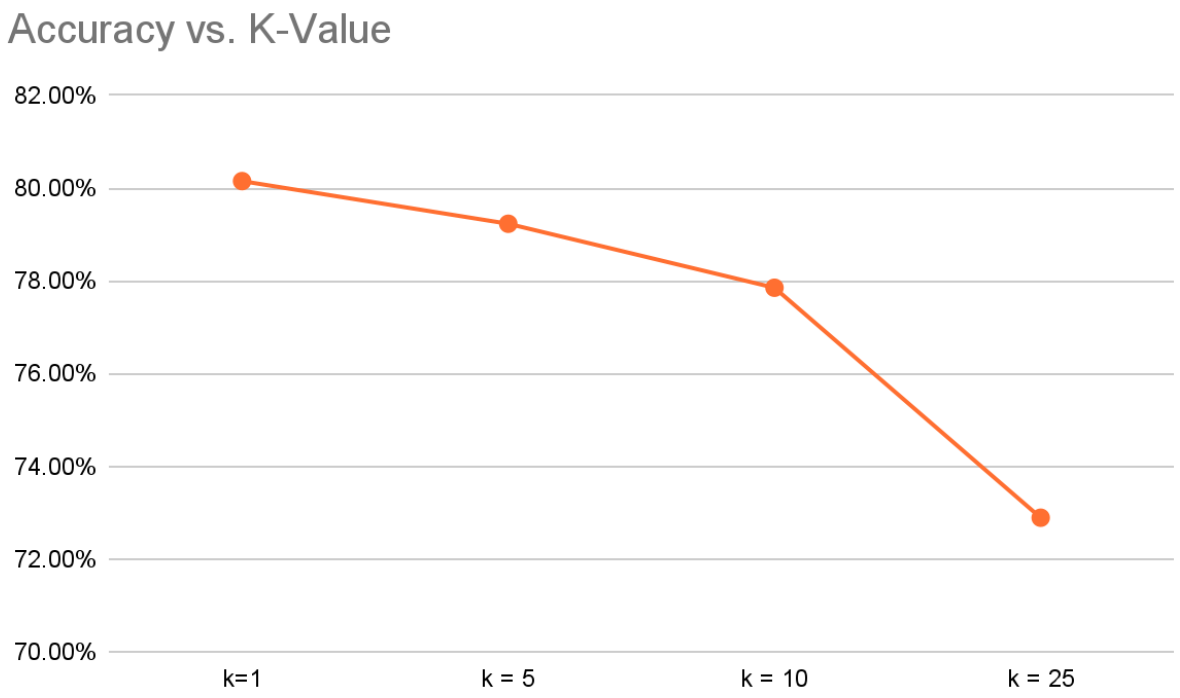
What kernel setting works best for the SVM model? (Linear, poly, or rbf)



Graph 2: comparison of f-scores for each SVM kernel type

Looking at the accuracies for each kernel setting, what I expected to perform best was actually what performed worst with this data set. I assume that this is due to the same reasoning given for the KNN error.

When using KNN algorithm, how do different K values affect the accuracy?



Graph 2: comparison of f-scores for each KNN k-value

It seems that when the k-value is increased with this data set, the accuracy of the KNN model goes down. I did some research using stack exchange, someone said that when using the same data set for training and testing it can lead to k=1 being the best fit.[6] This is the opposite of how the algorithm should have worked, the higher the k-value the better it performs in general.

[6] https://stackoverflow.com/questions/36637112/why-does-k-1-in-knn-give-the-best-accuracy

Lessons Learned

When using the same data set for training and testing it can lead to k=1 being the best fit, however it usually the greater k is, the better fit or accuracy. So, next time I would manually separate the data into two sets. Instead I relied on a built-in algorithm that may or may not truly separate the original data set for testing and training. I also learned to create my own KNN model to use for classification.

Acknowledgments

My largest help was from the developer who posted the data set, and discussed how to implement different models with this data set. The following are links to websites/tutorials I used to help with development of my experiment:

- https://machinelearningmastery.com/fbeta-measure-for-machine-learning/#:~:text=The%20F%2Dmeasure%20is%20calculated,model%20and%20in%20comparing%20models. For understanding and creating f-scores
- https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/ for answering why to undersample the training sets
- https://www.kaggle.com/code/meirnizri/covid-19-risk-prediction for ideas/insperation and help with KNN algorithm
- https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975 along with the class slides on Perceptron
- https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm/ for working with SVM
- https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html#pandas.read_csv for parsing through the data
- https://scikit-learn.org/stable/modules/neighbors.html used documentation for ML
- https://docs.scipy.org/doc/scipy/reference/spatial.distance.html?highlight=distance%20matrix used documentation for using distance algorithms
- https://stackoverflow.com/questions/36637112/why-does-k-1-in-knn-give-the-best-accuracy used to explain answer to investigation question