COVID-19 Rate Change Prediction

CS6375 - Machine Learning - Project Report

Submitted by :

Deepesh	FXD180002
Kapil Gautam	KXG180032
Vishwashrisairam Venkadathiriappasamy	VXV180043

Table of Contents

Problem description:

COVID-19 is a disease caused by a coronavirus called SARS-CoV-2. The first case of COVID-19 was reported in Wuhan, China in December 2019 and has since spread rapidly resulting in the ongoing Coronavirus pandemic. As of 2 May 2020, more than 3.42 million cases have been reported across 187 countries and territories, resulting in more than 243,000 deaths. More than 1.09 million people have recovered[1].

For this project, our goal is to study publically available datasets for the coronavirus pandemic and compare the predictions of different algorithms on these datasets. We have implemented and compared SVM and KNN algorithms. Further sections will describe in detail on the dataset and the observations/results for our experiments.

Initially we thought of using data available for previous disease outbreaks due to Coronaviruses (SARS, MERS etc.). But we decided not to do that as the SARS-CoV-2 outbreak has spread at an unprecedented level and very little data is available for other outbreaks when compared to COVID-19.

After collecting and analyzing the available data, we decided to define the problem as a classification problem by converting the features having continuous values to categorical values.

Dataset description:

One of the popular dataset available for Covid-19 is the time series data [3] tracking the number of people affected worldwide, which includes the number of confirmed cases, number of people who died while sick with Coronavirus and the number of people recovered from it. The data is disaggregated by country (and subregions sometimes) and is updated daily.

For this project however, we are using the dataset available from the Open Covid-19 project [4]. The repository contains the daily time series data related to Covid-19 for over 30 countries(including state/province data). It contains additional related data like the mobility data (provided by Google), weather data (daily weather information from the nearest station reported by NOAA.) and categorical data (number of new and current cases based on their severity level). For this project we'll be focusing only on the country "United States". We will consider state-wise data for our classification problem. However, we will not consider state as one of the features.

[1] "Coronavirus - Wikipedia." https://en.wikipedia.org/wiki/Coronavirus.

Our data preparation steps include collecting and merging the time series data with mobility, weather and categorical data and filtering out the rows for the United States.

Few of the rows in this data have missing values. These values are replaced with 0 since they could be interpreted as no data collected for that particular day. In the dataset, some of the features have continuous values. These were converted to categorical data since it is more suitable for classification tasks. All the features have been divided into following categories:

0 -> Very Low

1 ->Low

2 -> Medium,

3 -> High

4 -> Very High

We have tried to keep all the features balanced in terms of how many rows belong to a particular category. We were not able to do this for Deaths feature as there were a lot of rows with value 0 in the Deaths feature.

Additionally for the fields, Deaths and Confirmed we calculated the percentage change each day, since the spread was different in different states and the spread was much higher in states/places with large populations. Also the first death in the US was reported on February 28 2020. So our time series dataset consists of data beginning from February 28 to April 25 2020. There are a total 2958 rows in the dataset.

After removing the irrelevant features, the final features list in our dataset are:

['NewMild_cat', 'NewSevere_cat', 'NewCritical_cat', 'CurrentlyMild_cat', 'CurrentlySevere_cat', 'CurrentlyCritical_cat', 'RetailAndRecreation_cat', 'GroceryAndPharmacy_cat', 'Parks_cat', 'TransitStations_cat', 'Workplaces_cat', 'Residential_cat', 'DeathsRateChange_cat', 'ConfirmedRateChange_cat']

1. *'NewMild_cat'* - Number of estimated new mild cases from previous day
2. *'NewSevere_cat'* - Number of estimated new severe cases from previous day
3. *'NewCritical_cat'* - Number of estimated new critical cases from previous day
4. *'CurrentlyMild_cat'* - Number of estimated mild active cases at this date
5. *'CurrentlySevere_cat'* - Number of estimated severe active cases at this date
6. *'CurrentlyCritical_cat'* - Number of estimated critical active cases at this date

7. *'RetailAndRecreation_cat'* - Percentage change in visits to retail and recreation locations
8. *'GroceryAndPharmacy_cat'* - Percentage change in visits to grocery and pharmacy locations
9. *'Parks_cat'* - Percentage change in visits to park locations
10. *'TransitStations_cat'* - Percentage change in visits to transit station locations
11. *'Workplaces_cat'* - Percentage change in visits to workplace locations
12. *'Residential_cat'* - Percentage change in visits to residential locations
13. *'DeathsRateChange_cat'* - Percentage change in deaths from previous day
14. *'ConfirmedRateChange_cat'* - Percentage change in confirmed cases from previous day

We tested the performance of KNN and SVM algorithms implemented by us on the above dataset. For training we used 80% of the data and remainder of the data for testing. The classification task is predict the confirmed rate/death rate into one of the five categories:

0 -> Very Low

1 ->Low

2 -> Medium,

3 -> High

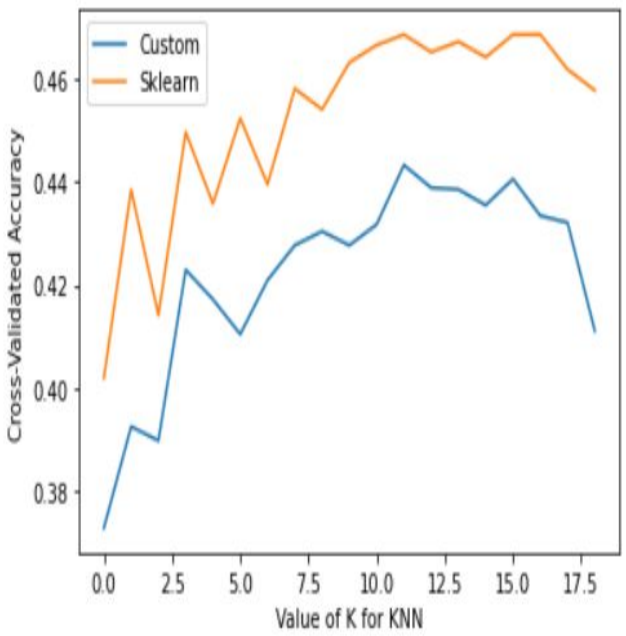
4 -> Very High

Further sections will describe the methodology and results of our experiments.

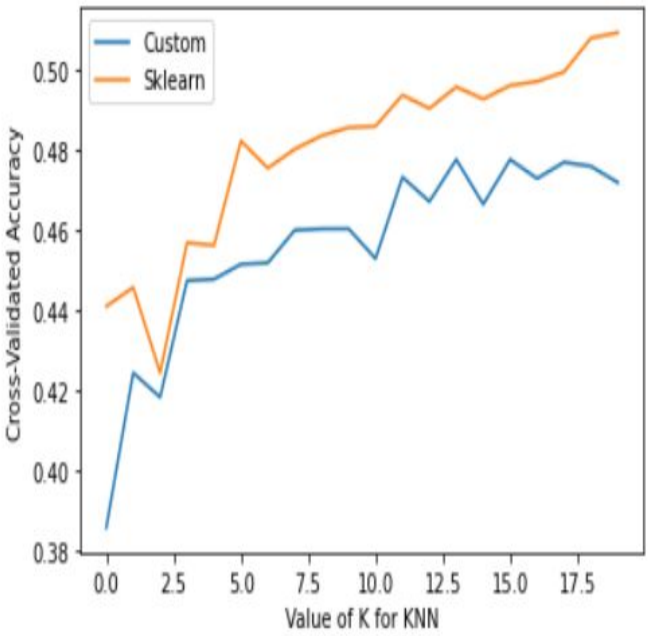
Custom KNN Algorithm Implementation:

The k-nearest neighbors algorithm is based around the idea that objects that are similar to each other will have similar characteristics. So any new instance can be classified by majority vote of its k nearest neighbors. The similarity between the data points is measured using the Euclidean distance metric. Since KNN is an online algorithm, it doesn't require specific training and the only hyperparameter for the model is the number of neighbors k.

The comparison of accuracy between our custom implementation and sklearn implementation for different values of k is shown in the graph below. The model is trained using 5 fold and 10 fold cross validation. As observed, the model performs best when the value of k is 12. The best accuracy is 47.31 % for 10 fold Cross Validation. The corresponding confusion matrix is also reported below:



Accuracy comparison: 5 fold cross validation



Accuracy comparison: 10 fold cross validation

One of the observations made while running the model for train-test split v/s n fold cross validation is that, for train-test split method, the best accuracy was obtained for k=7. Because of n fold cross validation, the overall variance will reduce and the model would perform better for unseen data. So after performing the cross validation, the best value is obtained at k= 12 as compared to the earlier method.

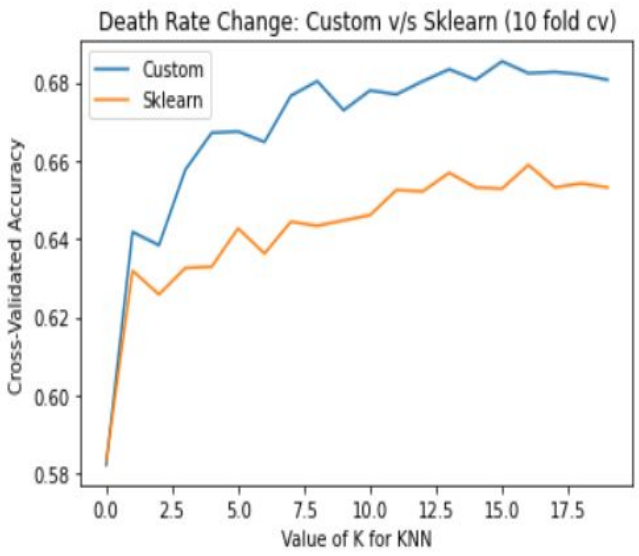
```
Accuracy: 0.5422297297297297
              precision    recall  f1-score   support

           0       0.72      0.90      0.80       126
           1       0.60      0.63      0.62       126
           2       0.39      0.42      0.40       110
           3       0.39      0.32      0.35       111
           4       0.51      0.39      0.44       119

    accuracy                           0.54       592
   macro avg       0.52      0.53      0.52       592
weighted avg       0.53      0.54      0.53       592

Confusion matrix:
[[113   0   1   0  12]
 [  1  80  32  11   2]
 [  3  37  46  15   9]
 [  7  12  34  36  22]
 [ 33   4   5  31  46]]
```

Confusion matrix for Confirmed Rate Change



Accuracy Comparison: For DeathRate Change

```
k=16
Accuracy: 0.6959459459459459
              precision    recall  f1-score   support

           0       0.81      0.94      0.87       297
           1       0.62      0.85      0.72       149
           2       0.00      0.00      0.00        57
           3       0.35      0.17      0.23        40
           4       0.12      0.02      0.04        49

    accuracy                           0.70       592
   macro avg       0.38      0.40      0.37       592
weighted avg       0.59      0.70      0.63       592

Confusion matrix:
[[278  11   1   1   6]
 [ 15 126   8   0   0]
 [ 11  39   0   7   0]
 [ 10  18   4   7   1]
 [ 31   9   3   5   1]]
```

Confusion matrix for Death Rate Change

One of the major pitfalls of using KNN is that as the data size increases it becomes computationally expensive to calculate the neighbors. Also it might not perform that well for high dimensional data or noisy/irrelevant data points.

Custom SVM Algorithm Implementation:

Support Vector Machine algorithm is chosen for these dataset as the size of data is not huge and there are 14 features for an example. SVM can be formulated as the dual problem, which allows us to use different kernels to our analysis. We are using the open source ‘CvxOpt’ mathematical library for calculating the alpha and the support vectors.

We have used the ‘linear’, ‘polynomial’, ‘sigmoid’ and ‘rbf’ kernels and the test accuracy of the model is comparable to the scikit-learn SVM implementation.

The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$. d is specified by keyword `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$. γ is specified by keyword `gamma`, must be greater than 0.
- sigmoid ($\tanh(\gamma \langle x, x' \rangle + r)$), where r is specified by `coef0`.

Figure : Reference scikit-learn documentation

The categorical data contains multiple classes which is inherently not supported by SVM. So we are using the 'One Vs Rest' classifier with SVM to generate a hyperplane for each category vs the rest, and then taking the maximum prediction category value for each example.

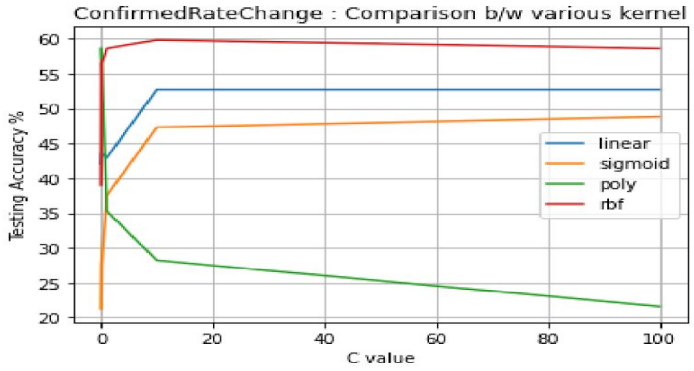
We have used multiple values of C : {0.01, 0.1, 1, 10, 100} to analyse the effect on the SVM in all the above mentioned kernels. Apart from all the kernel, we are reporting the confusion matrix for the best test accuracy for a given C value.

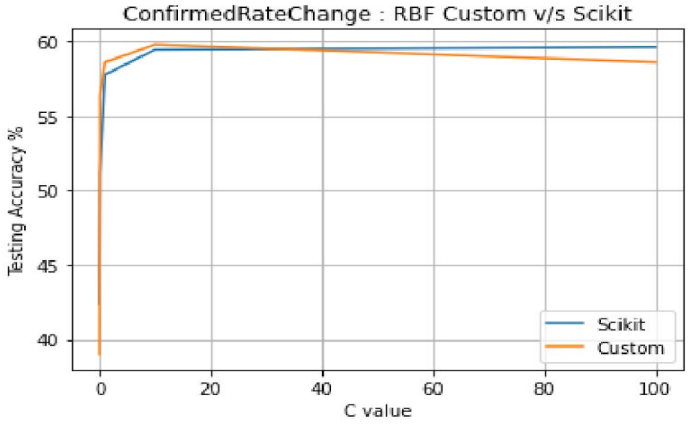
Issues with Custom SVM implementation : Currently the algorithm uses oneVsRest classification for the binary classifier SVM model. It has to run for each category against other classes which takes a long time to evaluate and is not scalable.

Possible solution : Use a Mutliclass SVM model and loss function to make the classifier fast and scalable.

Following are the graphs generated :

1. Confirmed Rate Change
    - Comparison b/w various Custom SVM implementation kernels
    - Comparison of RBF kernel b/w Custom and SciKit Learn implementation
    - Confusion Matrix for RBF kernel for Custom implementation, C = 10
2. Death Rate Change
    - Comparison b/w various Custom SVM implementation kernels
    - Comparison of Linear kernel b/w Custom and SciKit Learn implementation
    - Confusion Matrix for Linear Kernel for Custom implementation, C = 100





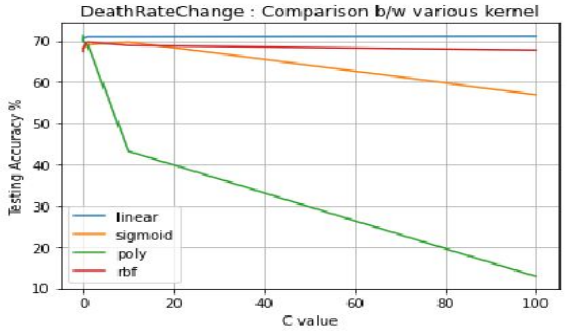
ConfirmedRate Change RBF Kernel, C=10

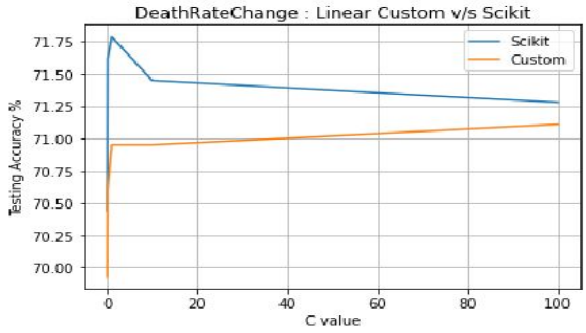
Accuracy : 59.80

	precision	recall	f1-score	support
0	0.75	0.88	0.81	126
1	0.72	0.63	0.68	126
2	0.51	0.46	0.49	110
3	0.43	0.45	0.44	111
4	0.53	0.52	0.53	119
accuracy			0.60	592
macro avg	0.59	0.59	0.59	592
weighted avg	0.60	0.60	0.59	592

Confusion Matrix:

```
[[111   0   0   1  14]
 [  2  80  23  19   2]
 [  2  27  51  20  10]
 [  6   4  22  50  29]
 [ 27   0   4  26  62]]
```





DeathRateChange Linear Kernel, C : 100

Accuracy : 71.11

	precision	recall	f1-score	support
0	0.79	0.96	0.87	297
1	0.63	0.84	0.72	149
2	0.29	0.18	0.22	57
3	0.00	0.00	0.00	40
4	0.00	0.00	0.00	49
accuracy			0.71	592
macro avg	0.34	0.40	0.36	592
weighted avg	0.59	0.71	0.64	592

Confusion Matrix:

```
[[286   8   3   0   0]
 [ 19 125   5   0   0]
 [ 12  35  10   0   0]
 [ 10  19  11   0   0]
 [ 33  10   6   0   0]]
```

Performance Evaluation:

```
Accuracy: 0.5422297297297297
              precision    recall  f1-score   support

           0       0.72      0.90      0.80       126
           1       0.60      0.63      0.62       126
           2       0.39      0.42      0.40       110
           3       0.39      0.32      0.35       111
           4       0.51      0.39      0.44       119

    accuracy                           0.54       592
   macro avg       0.52      0.53      0.52       592
weighted avg       0.53      0.54      0.53       592

Confusion matrix:
[[113   0   1   0  12]
 [  1  80  32  11   2]
 [  3  37  46  15   9]
 [  7  12  34  36  22]
 [ 33   4   5  31  46]]
```

ConfirmedRateChange using kNN with k=12

ConfirmedRate Change RBF Kernel, C=10

```
Accuracy : 59.80
              precision    recall  f1-score   support

           0       0.75      0.88      0.81       126
           1       0.72      0.63      0.68       126
           2       0.51      0.46      0.49       110
           3       0.43      0.45      0.44       111
           4       0.53      0.52      0.53       119

    accuracy                           0.60       592
   macro avg       0.59      0.59      0.59       592
weighted avg       0.60      0.60      0.59       592

Confusion Matrix:
[[111   0   0   1  14]
 [  2  80  23  19   2]
 [  2  27  51  20  10]
 [  6   4  22  50  29]
 [ 27   0   4  26  62]]
```

ConfirmedRateChange using SVM with C=10

DeathRateChange Linear Kernel, C : 100

```
Accuracy : 71.11
              precision    recall  f1-score   support

           0       0.79      0.96      0.87       297
           1       0.63      0.84      0.72       149
           2       0.29      0.18      0.22        57
           3       0.00      0.00      0.00        40
           4       0.00      0.00      0.00        49

    accuracy                           0.71       592
   macro avg       0.34      0.40      0.36       592
weighted avg       0.59      0.71      0.64       592

Confusion Matrix:
[[286   8   3   0   0]
 [ 19 125   5   0   0]
 [ 12  35  10   0   0]
 [ 10  19  11   0   0]
 [ 33  10   6   0   0]]
```

DeathRateChange SVM with linear Kernel

```
k=16
Accuracy: 0.6959459459459459
              precision    recall  f1-score   support

           0       0.81      0.94      0.87       297
           1       0.62      0.85      0.72       149
           2       0.00      0.00      0.00        57
           3       0.35      0.17      0.23        40
           4       0.12      0.02      0.04        49

    accuracy                           0.70       592
   macro avg       0.38      0.40      0.37       592
weighted avg       0.59      0.70      0.63       592

Confusion matrix:
[[278  11   1   1   6]
 [ 15 126   8   0   0]
 [ 11  39   0   7   0]
 [ 10  18   4   7   1]
 [ 31   9   3   5   1]]
```

DeathRateChange kNN with k=16

SVM gives better accuracy than kNN in both the cases. One of the reasons could be the number of dimensions. kNN does not work well for high dimensional data. Another reason could be irrelevant or non-correlated features.

Another interesting observation is both the models have a good F1 score for lower values of rate change. Both the models are not able to correctly predict for higher values of rate change. One of the reasons could be there is relatively less data and hence the model is not able to generalize for those categories. This is also the same for other classifiers as shown in the next section.

Other Classifiers:

Apart from the classifiers discussed on the above sections, we tried running other models available in the scikit-learn library. The performance of those models are tabulated as below:

Classifier	**Accuracy (Confirmed Rate)**	**Accuracy (Death Rate)**
KNN*	54.22%	69.59%
SVM*	59.80%	71.11%
Logistic regression	56.75%	70.44%
Random Forest	59.12%	70.77%
XgBoost	59.79%	72.12%

Further improvements:

Currently, we are only considering the daily rate change in deaths and confirmed cases. One simple improvement could be to consider 3-day or 7-day rate change. Then, we will be able to predict 3-day or 7-day rate change for the future.

We have the temperature and Government response data available as well. These datasets could be incorporated into the dataset that we have prepared.

Currently we are only considering the data for the United States. We could possibly extend it to other countries as well.

References:

1. CVXOPT library : https://courses.csail.mit.edu/6.867/wiki/images/a/a7/Qp-cvxopt.pdf
2. Mathematical Formulation : https://scikit-learn.org/stable/modules/svm.html#mathematical-formulation
3. Dataset : https://github.com/datasets/covid-19
4. Dataset : https://github.com/open-covid-19/data