The COVID-19 Health Risk Prediction and Modeling for USA Counties

Tingting Huang

Fall 2020 | Term Project

CPLN 670 Geospatial Software Design

CONTENT

INTRODUCTION

Many studies have indicated that the dynamics and spread of COVID-19 incidence has a correlation with geo environmental and demographic characteristics of a specific region. A study suggests a strong positive relationship of COVID-19 incidence with the median household income, income inequality, percentage of nurse practitioners, and percentage of black female (Abolfazl Mollalo, Behzad Vahedi et al., 2020). Additionally, some studies from China indicate that the meteorological factors also influence COVID-19 spread, potentially with an interactive effect between daily temperature and relative humidity on COVID-19 incidence. The risk of an increased daily incidence of SARS was 18.18 times greater at lower temperature. (Hongchao Qi, Shuang Xiao et al., 2020). As temperature fall, winter could bring a new challenge for COVID-19 spreading. The project aims to show the spatial-temporal distribution characteristics of COVID-19 incidence in USA from January to December, as well as training a random forest classifier to predict the health risk to COVID-19 for each county. Additionally, the project explores the correlations between the demographic, social-economic and environment variables with the risk level to COVID-19, and the factors that contribute most to the COVID-19 risk increase for counties from a geospatial perspective.

DATA

The daily count of confirmed COVID-19 Cases for each county in USA were collected from the official report that is released by Coronavirus Resource Center of Johns Hopkins University(https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series). Since the first Covid-19 Case in US was reported in Jan 22, 2020, the confirmed cases in the 22nd of each month from January to November were used to show the variation trend of COVID-19 in the project. The demographic and social-economic data, including total population, population in different race and age groups, median income, housing unit, unemployment rate, population commuting by public transportation, were collected from 2018 American Community Survey 5-Year Estimates dataset. The mask-use frequency data was retrieved from Google Big Query public dataset. The meteorological data, including daily mean 2 m air temperature and daily total precipitation sums of each county are available in ERA5-Land dataset, since the meteorological data are reported daily, they were retrieved and converted to annual mean temperature and annual mean precipitation for each county in Google Earth Engine. The US county geographical data was collected from United States Census Bureau (https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html)

The data were merged and processed using Arcpy.

TEMPERATURE AND PRECIPITATION DATA PROCESSING

• Import the daily temperature and precipitation data from July 2019 to July 2020 from the ERA5-Land dataset.

```
// Import the feature collection
var us_county  = ee.FeatureCollection( 'users/huangtt/US_County' );
var us_outline  = ee.FeatureCollection( 'users/huangtt/county_dissolve' );

// Create a feature for US boundary
var us = us_outline.filter(ee.Filter.eq("Feature Index  ", '0'));

// Daily mean 2m air temperature
var temperature = ee.ImageCollection('ECMWF/ERA5/DAILY')
                  .select('mean_2m_air_temperature')
                  .filter(ee.Filter.date('2019-07-01', '2020-07-01'));

// Daily total precipitation sums
var precipitation = ee.ImageCollection('ECMWF/ERA5/DAILY')
                  .select('total_precipitation')
                  .filter(ee.Filter.date('2019-07-01', '2020-07-01'));
```

• Clip the layers with US boundary and calculate the annual mean temperature and annual mean precipitation for the new defined image.

```
// Clip the layer to only include US
var us_tp = ee.ImageCollection(temperature).map(function(image){return image.clip
(us_county)});
var us_pp = ee.ImageCollection(precipitation).map(function(image){return image.cl
ip(us_county)});

// Calculate the average temperature and average precipitation over the year
var us_tp_mean_k = us_tp.mean();
var us_pp_mean = us_pp.mean();

// Transfer the temperature unit from Kelvin to Celsius
var us_tp_mean = us_tp_mean_k.subtract(273.15);
```

• Create a function to calculate the mean value of all grids inside the boundary for each county and apply the function to the mean temperature image, export the data.

```
// Create a function to calculate the average annual temperature for each county
// Then apply the function to all US counties and export the table to drive
var clip_tp = function(feature, list){
```

```
    var tp_county = us_tp_mean.clip(feature);
    var geom = feature.geometry();
    var mean_tp_county = tp_county.reduceRegion({
      reducer: ee.Reducer.mean(),
      geometry: geom,
      scale: 1000,
      bestEffort: true});
    var tp = mean_tp_county.get('mean_2m_air_temperature');
    return feature.set('temperature', tp);
  };

  var tp_county = us_county.map(clip_tp);

  Export.table.toDrive({
    collection: tp_county,
    fileFormat: 'CSV',
  });
```

• Create a function to calculate the mean value of all grids inside the boundary for each county and apply the function to the mean precipitation image, export the data to Google Drive.

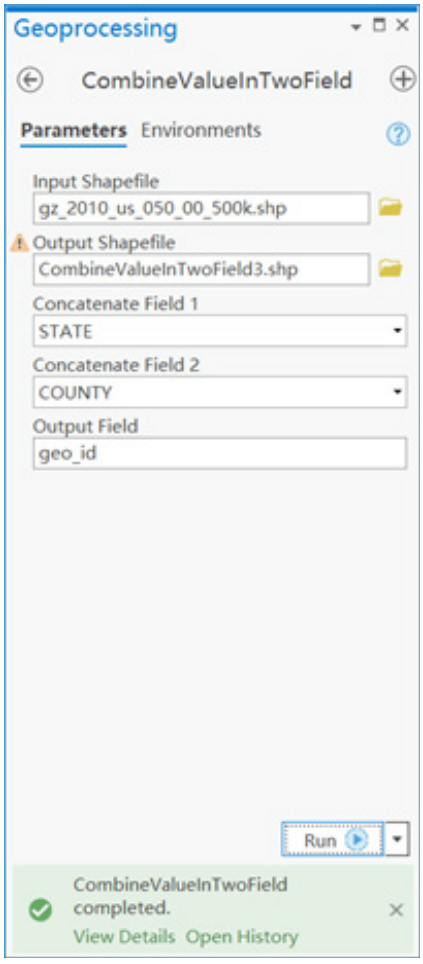
```
// Create a function to calculate the total precipitation for each county
// Then apply the function to all US counties and export the table to drive
var clip_pp = function(feature, list){
    var geom = feature.geometry();
    var mean_pp_county = us_pp_mean.reduceRegion({
      reducer: ee.Reducer.mean(),
      geometry: geom,
      scale: 1000,
      bestEffort: true});
    var pp = mean_pp_county.get('total_precipitation');
    return feature.set('total_precipitation', pp);
  };
  var pp_county = us_county.map(clip_pp);

  Export.table.toDrive({
    collection: pp_county,
    fileFormat: 'CSV',
  });
```

DEMOGRAPHIC AND SOCIOECONOMIC DATA PROCESSING

Three tools were created in Arcpy to process the demographic and social-economic data.

1. Combine Value in Two Fields

There is no ready-made FIPS code in the county shapefile, but the state code and county code are available in the county shapefile. To merge the demographic and social-economic data, the tool was created to combine the string in state code and county code field. The code is available in the appendix.



Tool User Interface

STATE	COUNTY
01	029
01	031
01	037
01	039
01	041
01	045
01	049
01	053
01	057
01	061
01	067

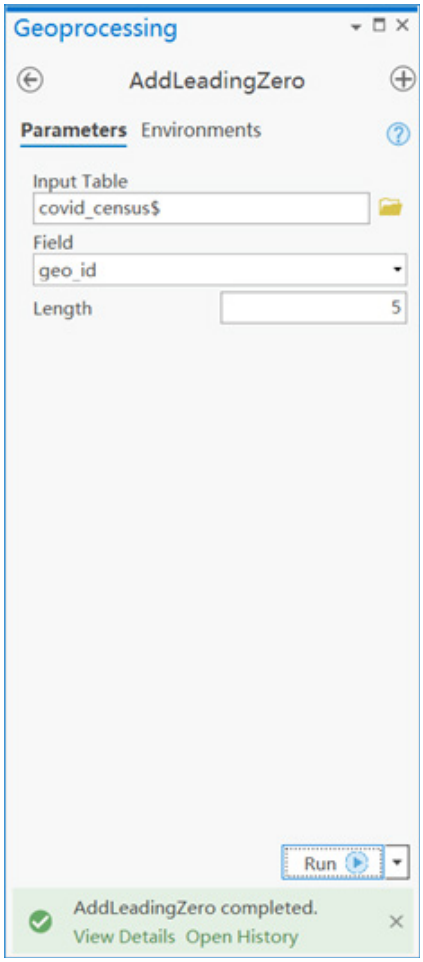
Input

FIPS
01029
01031
01037
01039
01041
01045
01049
01053
01057
01061
01067

Output

2. Add Leading Zero

The FIPS codes of demographic and social-economic data are in integer format, the leading zero of some counties' FIPS code are disappear, the Add Leading Zero tool was created to make the FIPS code of all county in social-demographic dataset coincident with the county shapefile's FIPS code. The code is in appendix.



Tool User Interface

	FIPS	geo_id_1	Admin2
0	1029	1029.0	Cleburne
2	1031	1031.0	Coffee
5	1037	1037.0	Coosa
5	1039	1039.0	Covington
0	1041	1041.0	Crenshaw

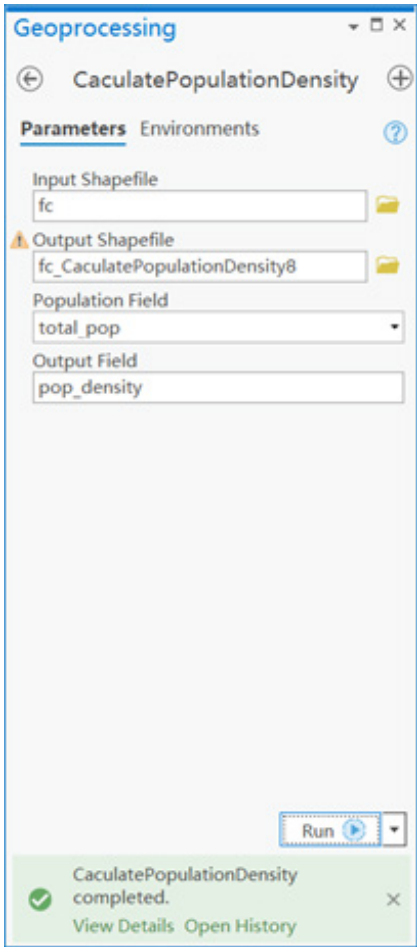
Input

geo_id	
01029	C
01031	C
01037	C
01039	C
01041	C
01045	D
01049	D
01053	E
01057	F
01061	G
01067	H

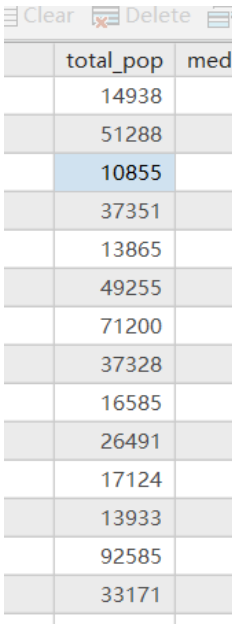
Output

3. Calculate Population Density

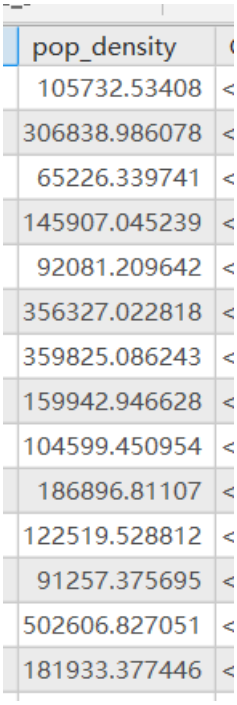
The tool can calculate the population density based on the inputted population data. The code is available in the appendix.



Tool User Interface



Input



Output

METHODOLOGY

The project performs the Covid-19 risk prediction using random forest model. Random is an ensemble learning method for classification, regression which operate by trains multiple decision trees in parallel in training dataset, each split at the nodes of the trees is determined only by a random subset of features, then aggregate the votes of decision trees to decide the class label of test dataset.

Before implementing the random forest classifier, the risk for COVID-19 in each county was graded according to the growth rate of COVID-19 case during last three months, and number of confirmed cases. The evaluation criteria are as below. Then, I created an ArcGIS Pro tool to perform the classification for all the county, the code is in the appendix.

COVID-19 Cases Growth Rate = Confirmed Cases in November -Confirmed Cases in September

COVID-19 Risk Classification Cariteria

COVID-19 Cases Growth Rate Change	Confirmed COVID-19 Cases	Risk Class	Count
<50	<1000	1	651
>=50	<1,000	2	1080
	>=1,000 and <4,000	3	1003
<= 150	>=10,000	4	305
	>=4,000 and <10,000		
>150	10,000	5	244

RANDOM FOREST CLASSIFICATION

The Random Forest was implemented using Arcpy and the scikit learn library.

• Import libraries

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, confusion_matrix,plot_confusion_m
atrix

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


import arcpy as ARCPY
import arcpy.da as DA
import arcgisscripting as ARC
import SSUtilities as UTILS
import os as OS
```

• Name the feature class that contains the features that will be used in prediction.

```
inputFC = r'fc_df'
```

• Define the names of prediction variables (Demographic, socioeconomic, meteorological features) and the classification variable (COVID-risk).

```
# Features
predictVars = [
 'pop_density',
 'median_income',
 'vacant_housing_units',
 'dwellings_50_or_more_units',
 'gini_index',
 'pct_pop_under18',
 'pct_pop_over65',
 'pct_white_pop',
 'pct_black_pop',
 'pct_asian_pop',
 'pct_hispanic_pop',
 'pct_unemployed_pop',
 'mask_always',
```

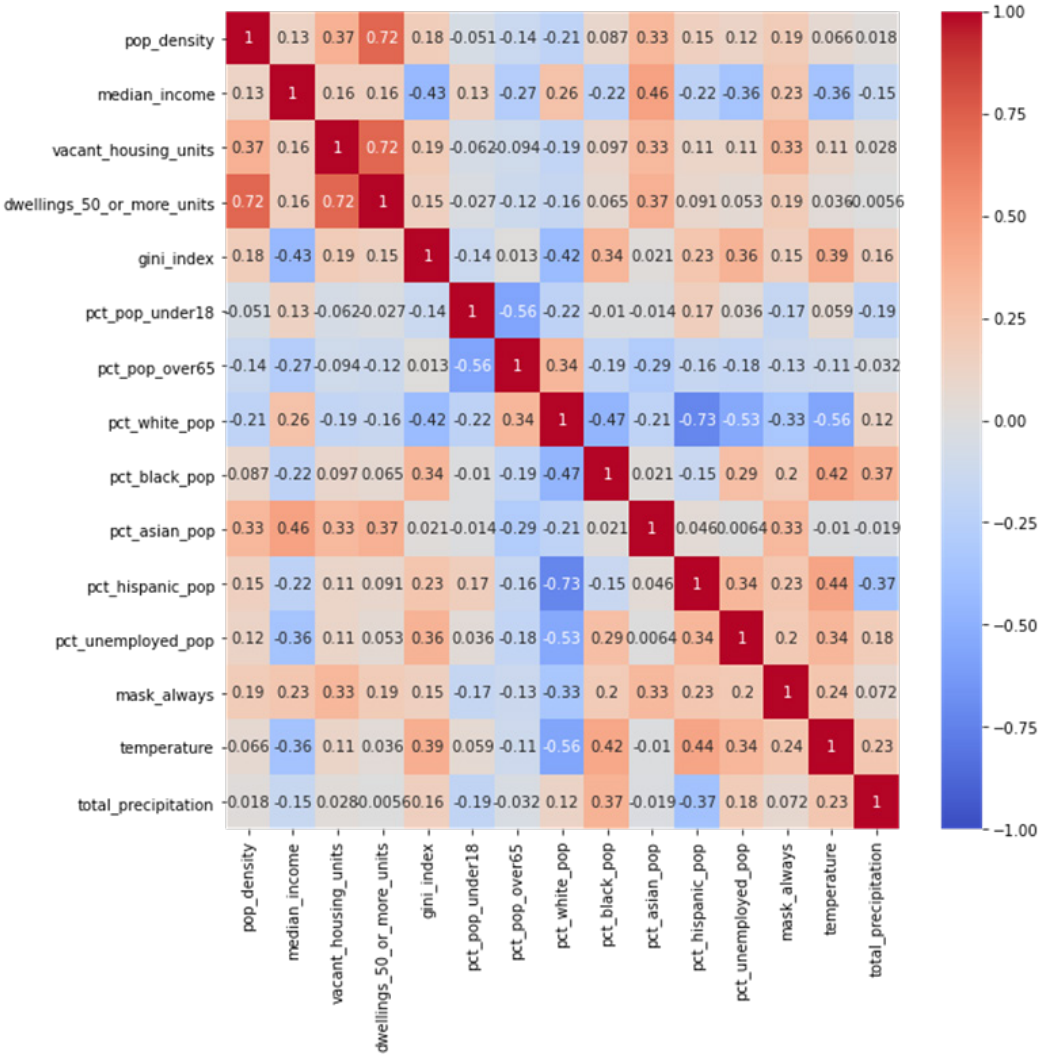
```
 'temperature',
 'total_precipitation']
classVar = ['covid_risk_class']
allVars = predictVars + classVar
```

• Read in feature class data and set the spatial reference

```
# Set spatial reference
trainFC = DA.FeatureClassToNumPyArray(inputFC, ["SHAPE@XY"] + allVars)
spatRef = ARCPY.Describe(inputFC).spatialReference
# Convert to pandas dataframe
data = pd.DataFrame(trainFC, columns = allVars)
```

• Plot the correlation coefficients as a correlation matrix between variables

```
# Check correlation
corr = data.astype('float64').corr()
ax = sns.heatmap(corr, cmap='coolwarm', annot=True, vmin=-1, vmax=1, ax=ax)
plt.show()
```



• Create training and testing dataset.

```
# Create training and testing dataset
fracNum = 0.3
train_set = data.sample(frac = fracNum)
test_set = data.drop(train_set.index)
print('Training Data Size = ' + str(train_set.shape[0]))
print('Test Data Size = ' + str(test_set.shape[0]))
```

• Standardize prediction variables and initialize a random forest model pipeline.

```
# Set up the column transformer with two transformers
transformer = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), num_cols),
    ]
)

# Initialize the random forest model pipeline
rfclf = make_pipeline(
    transformer, RandomForestClassifier(n_estimators=100, random_state=42)
)
```

• Train the random forest classifier using the training data. Print the accuracy score.

```
# Train a model
rfclf.fit(train_set[predictVars], train_set[classVars])
riskPred = rfclf.predict(test_set[predictVars])

# Print the training score
training_score = rfclf.score(train_set[predictVars], train_set[classVars])
print(f"Training Score = {training_score}")

# Print the test score
test_score = rfclf.score(test_set[predictVars], test_set[classVars])
print(f"Test Score = {test_score}")

# Print the MSE
mse_test = mean_squared_error(y_test, y_pred)
print("mse of random forest is: ", mse_test)
```

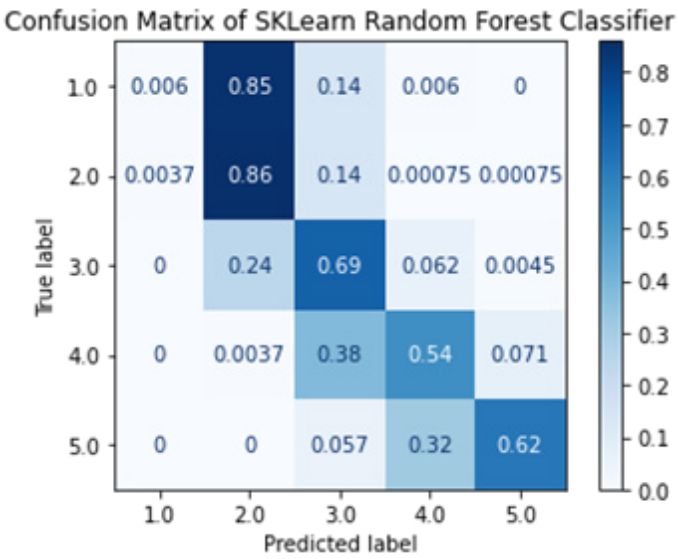
```
Random forest
Training Score = 1.0
Test Score = 0.711425612702796
mse of random forest is:  0.33759061097687265
```

Accuracy Score and MSE

• Plot the confusion matrix and the importance of each prediction variable.

```
# plot confusion matrix
disp = plot_confusion_matrix(rfclf, test_set[predictVars], test_set[classVars],cm
ap=plt.cm.Blues, normalize='true')
disp.ax_.set_title('Confusion Matrix of SKLearn Random Forest Classifier')

plt.show()
```

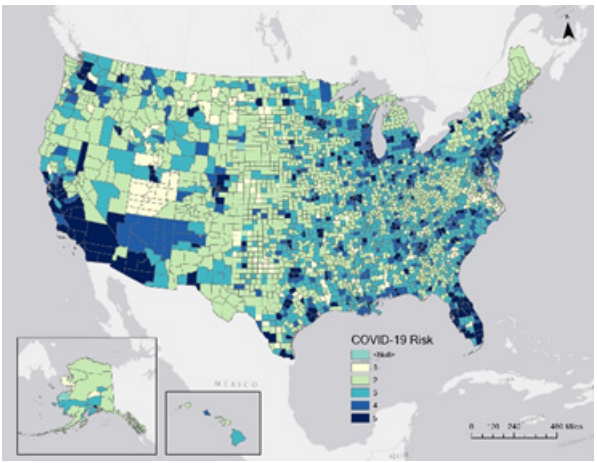
```
# The classifier
classifier = forest_pipe["randomforestclassifier"]

# Full list of columns is numerical + one-hot
features = num_cols

# Create the dataframe with importances
importance = pd.DataFrame(
    {"Feature": features, "Importance": classifier.feature_importances_}
)

# Sort importance in descending order and get the top
importance = importance.sort_values("Importance", ascending=False).iloc[:30]

# Plot
importance.hvplot.barh(
    x="Feature", y="Importance", flip_yaxis=True, height=500
)
```



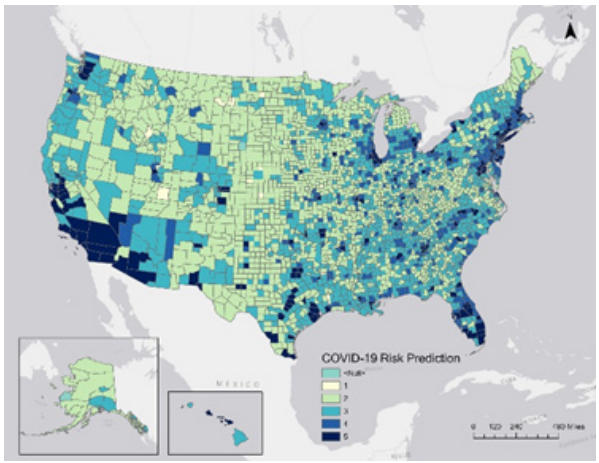
RESULTS

COVID-19 RISK PREDICTION

The final model produces a mean squared error of 0.337and an accuracy score of 0.711. The random forest classifier has a good performance in predicting counties with high COVID-19 risk level but cannot classify the counties with less risk for COVID-19 based on their demographical, socioeconomic, and meteorological features very well. The model also underpredict the risk level of many counties in the Midwest. From the COVID-19 risk map, we can see that the counties with highest COVID-19 risk are in the metropolitan areas. Many counties in the Midwest, Alaska have less COVID-19 risk.

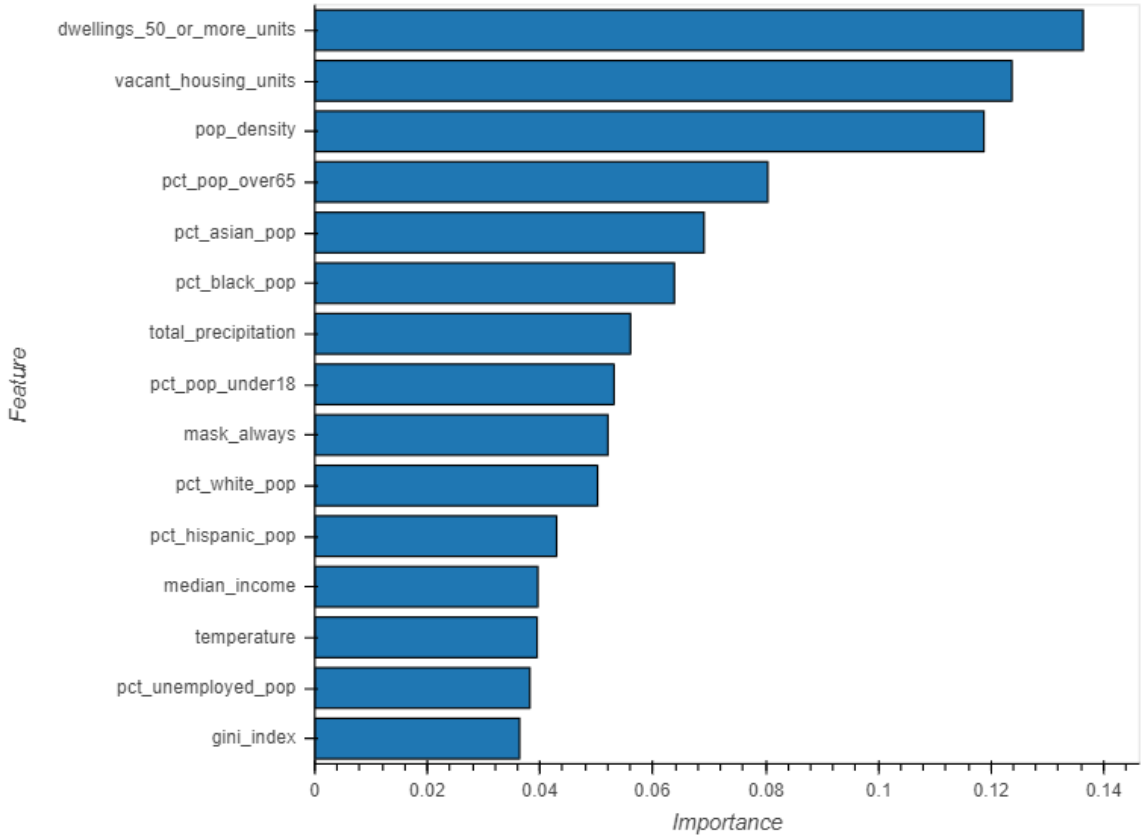


COVID-19 Risk



COVID-19 Risk prediction

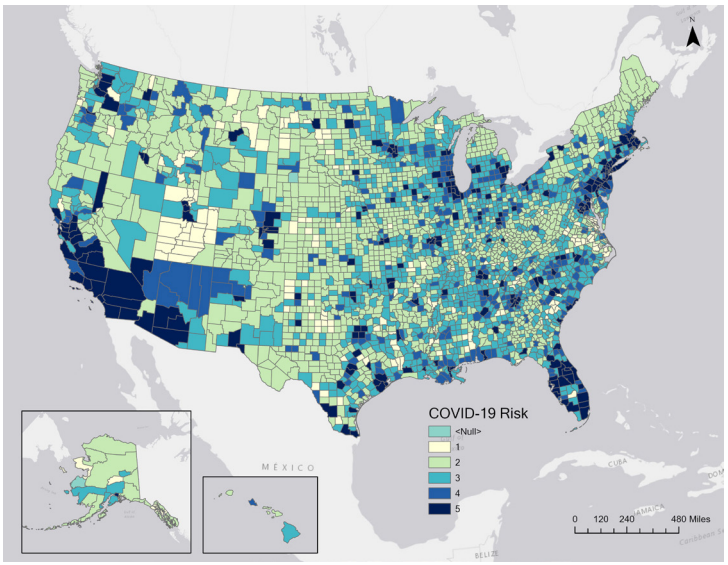
From the firgure below, we can see that the residential density, population density, ration of elderly population, ratio of population in ethnic minority, and precipitation, the percentage of residents always putting mask on plays an important role in predict the COVID-19 risk level.

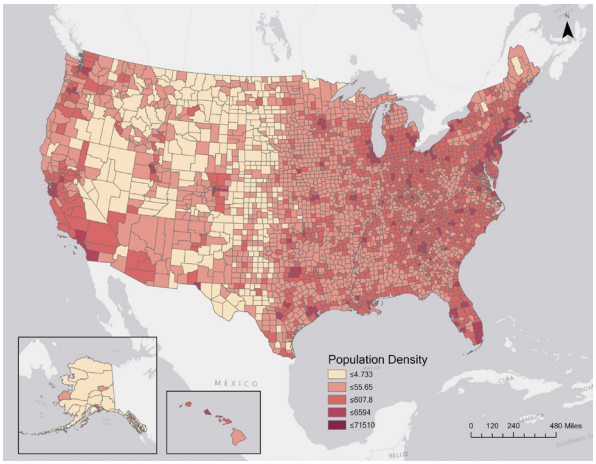


Densely-populated counties are more likely to be hit severely by the pandemic, these counties are facing higher COVID-19 health risk.

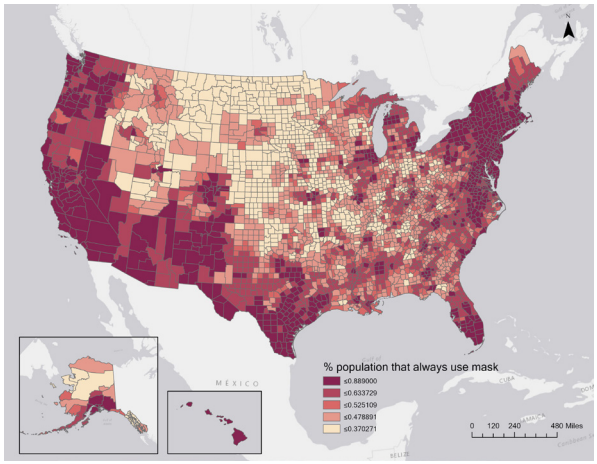
Additionally, there is higher percetage of population awalys put mask on in counties with High COVID-19 risk. The counties with a higher ratio of aged population are more likely to have a higher COVID-19 risk, for the counties in West coast area, the aged population is not the risk source for COVID-19.

COVID-19 Risk

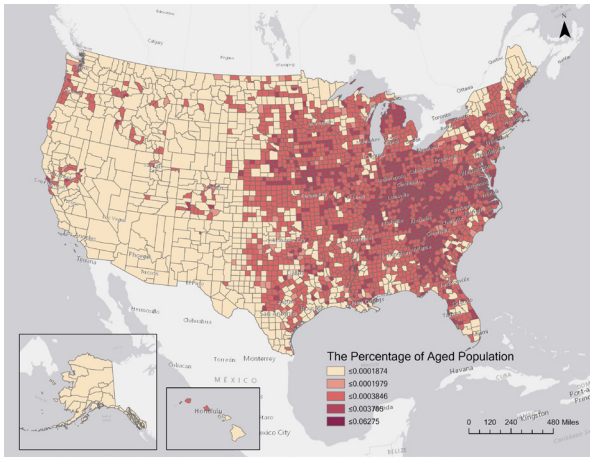




Population Density



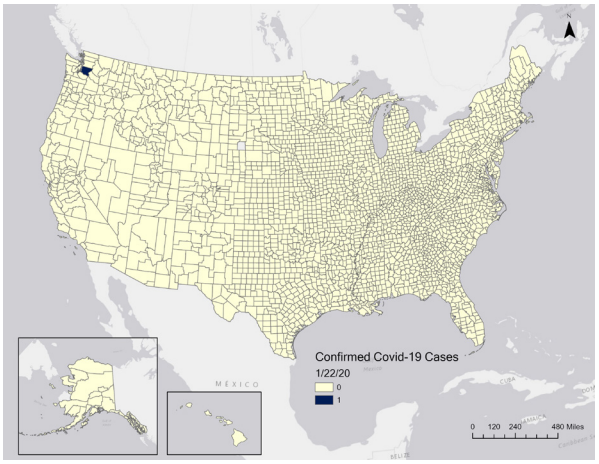
% Population that
Always Putting Mask on



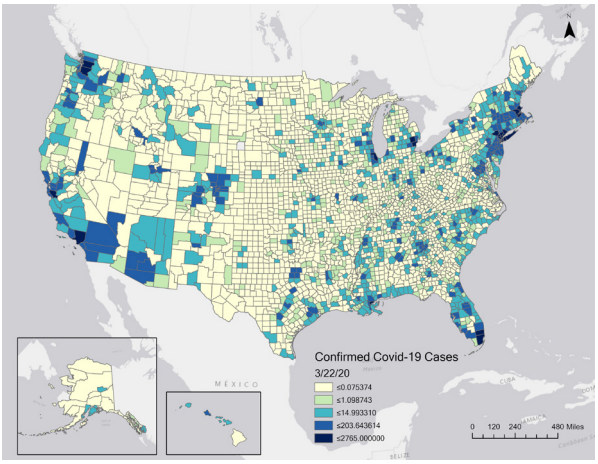
% Aged Population

COVID-19 SPATIAL-TEMPORAL EVOLUTION AND TEMPERATURE

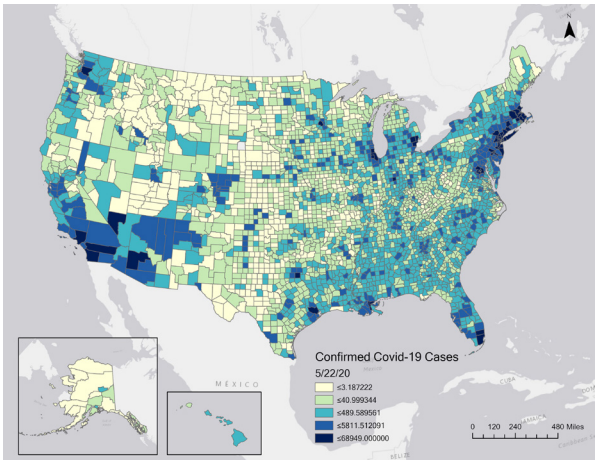
COVID-19 SPATIAL-TEMPORALDISTRUBTION



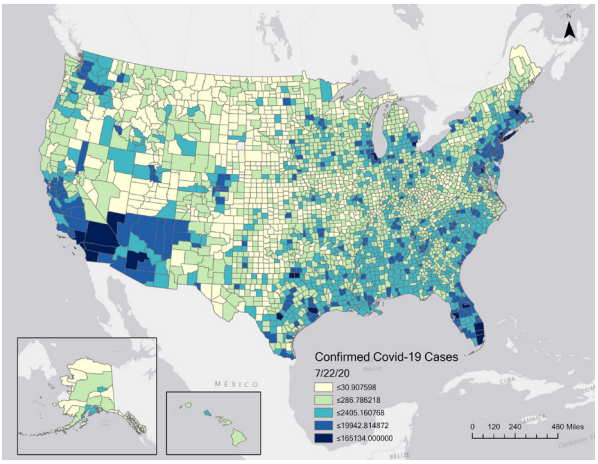
Jan



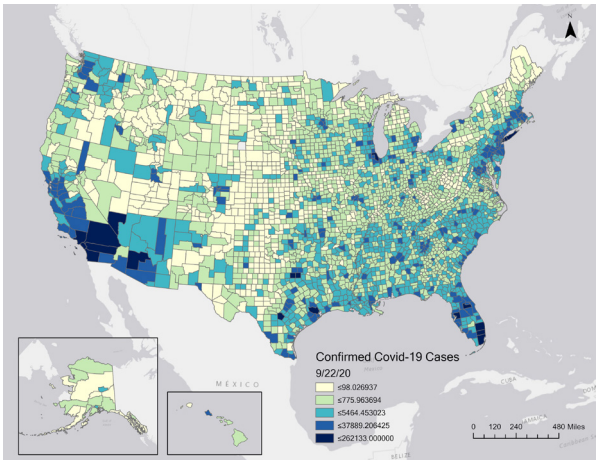
Mar



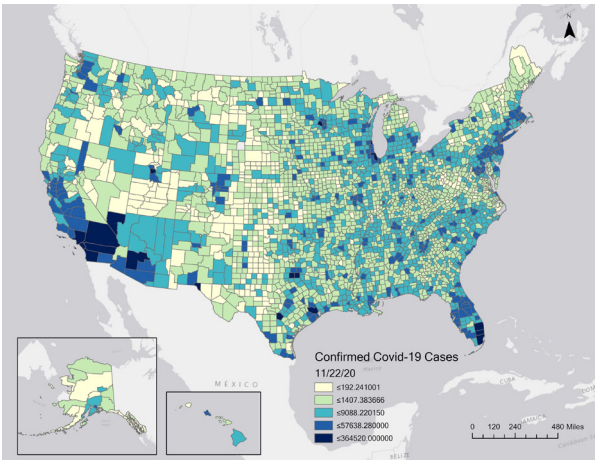
May



Jul

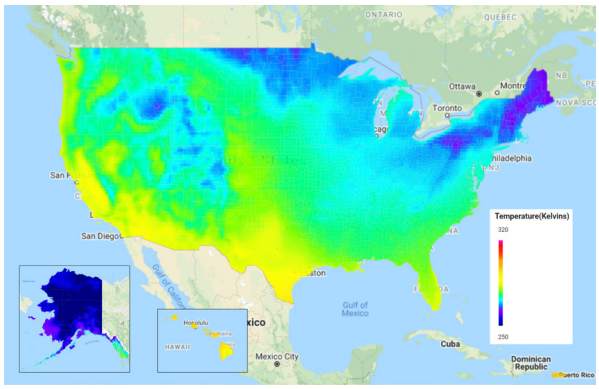


Sep

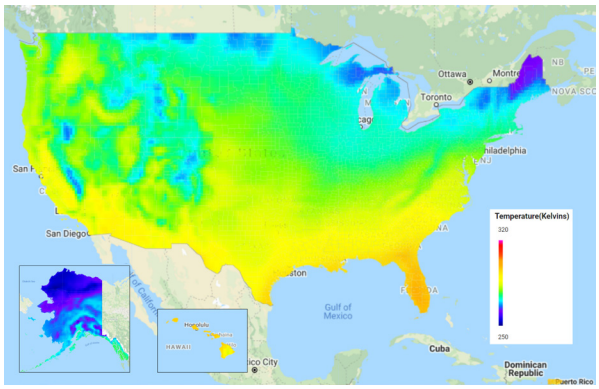


Nov

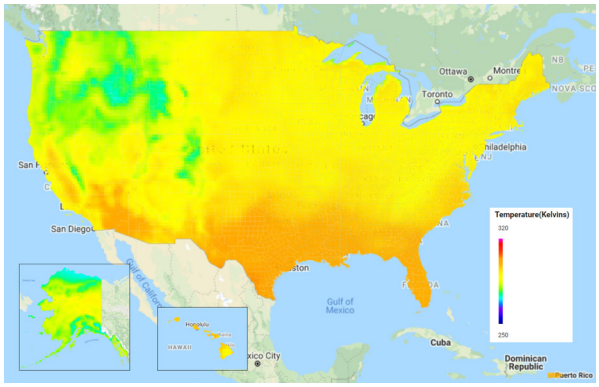
TEMPERATURE CHANGE



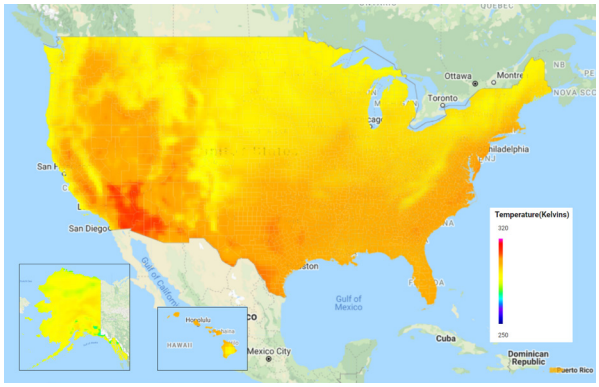
Jan



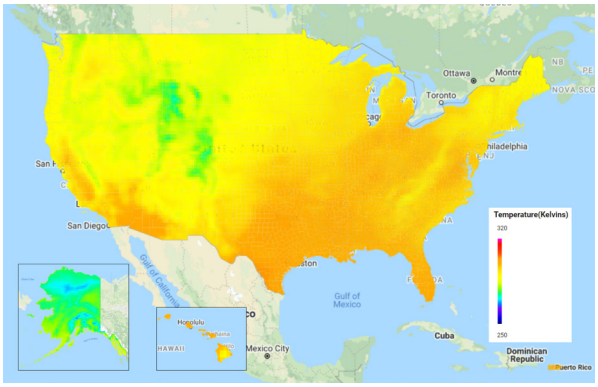
Mar



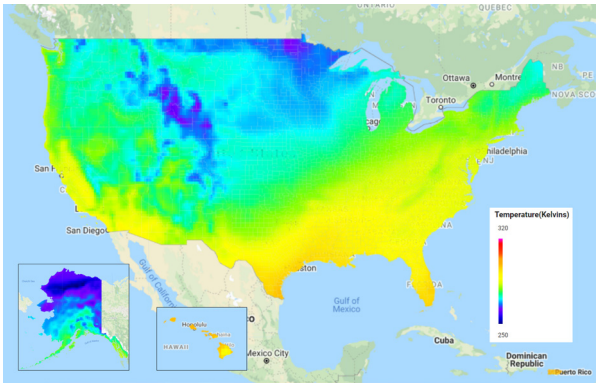
May



Jul



Sep



Nov

CONCLUSION

Demographic, social-economic factors, including population density, residential density, the ratio of aged population, the ratio of population that always putting mask on, and meteorological factors influence COVID-19 transmission and spread. These factors can be used to predict the COVID-19 risk the US counties facing. The random forest classifier constructed based on these prediction variables tends to overpredict the Midwest counties with less COVID-19 risk.

Population density is a very important factor in COVID-19 risk prediction. The counties with high number of COVID-19 cases and high COVID-19 cases growth rate are mainly concentrated in densely populated urban areas. Additionally, the residual density also plays an important role in COVID-19 risk prediction. The counties with more housing that more than 50 units are facing higher COVID-19 health risk since it is more difficult to more likely to expose to the virus when more people share housing facilities such as elevator. Therefore, it is essential to always put mask on and keep social distancing in areas with high population density and residential density.

Moreover, the percentage of population age 65 and older and the ratio of ethnic minority population are also important indicate the counties with high COVID-19 risk. It is consistent with the research made by CDC showing that the COVID-19 death rate of population age 65 and older is 90 times higher than the young. Black people, Hispanic people, and non-Hispanic American Indian or Alaska Native people have a COVID-19 hospitalization rate over 2.5 times that of non-Hispanic white people. The disparity in races reflects the underlying socioeconomic status. Many people of color have jobs that cannot be done remotely and involve interaction with the public, increasing exposure to the virus.

REFERENCE

Franch-Pardo, I., Napoletano, B. M., Rosete-Verges, F., & Billa, L. (2020). Spatial analysis and GIS in the study of COVID-19. A review. Science of The Total Environment, 140033.

Mollalo, A., Vahedi, B., & Rivera, K. M. (2020). GIS-based spatial modeling of COVID-19 incidence rate in the continental United States. Science of The Total Environment, 138884. Qi, H., Xiao, S., Shi, R., Ward, M. P., Chen, Y., Tu, W., ... & Zhang, Z. (2020).

COVID-19 transmission in Mainland China is associated with temperature and humidity: A time-series analysis. Science of the Total Environment, 138778.

Emsi County Health Risk Index: https://www.economicmodeling.com/2020/06/04/county-health-risk-index/

COVID-19 Hospitalization and Death by Age: https://www.cdc.gov/coronavirus/2019-ncov/covid-data/investigations-discovery/hospitalization-death-by-age.html

COVID-19 Hospitalization and Death by Race/Ethnicity: https://www.cdc.gov/coronavirus/2019-ncov/covid-data/investigations-discovery/hospitalization-death-by-race-ethnicity.html

Does Density Aggravate the COVID-19 Pandemic?: https://www.tandfonline.com/doi/full/10.1080/01944363.2020.1777891

APPENDIX

Code for Combine Value in Two Fields Tool

```
"""
THIS SCRIPT AIMES COMBINE TEXT IN TWO FIELDS INTO ONE AND ADD A NEW FIELD.
To create an ArcGIS Pro script tool for this script, do the following.
1   In Catalog > Toolboxes, select an existing toolbox or create a new one.
2   Right-click on the entry for this toolbox in ArcToolbox, and use New > Script to open a dialog box.
3   In this dialog box, use Label to name the tool being created and Script File to specify its .py file.
        LABEL/NAME          DATA TYPE           DIRECTION
        Input Shapefile?    Shapefile           Input
        Output Shapefile?   Shapefile           Output
        Concatenate Field 1 Field               Input
        Concatenate Field 2 Field               Input
        Output Field?       Field               Input
4   To later revise any of this, right-click to the tool's name and select Properties.
"""

# Import necessary modules
import sys, os, string, math, arcpy, traceback

# Allow output file to overwrite any existing file of the same name
arcpy.env.overwriteOutput = True

try:

    # Request user input of data type = Shapefile and direction = Input
    InputShapefile = arcpy.GetParameterAsText(0)
    arcpy.AddMessage('\n' + "The input shapefile name is " + InputShapefile)

    # Request user input of data type = Shapefile and direction = Output
    OutputShapefile = arcpy.GetParameterAsText(1)
    arcpy.AddMessage("The output shapefile name is " + OutputShapefile)

    # Request user input of data type = String and direction = Input
    Field1 = arcpy.GetParameterAsText(2)
    arcpy.AddMessage("The name of the field 1 to be combined is " + Field1 + "\n")

    # Request user input of data type = String and direction = Input
    Field2 = arcpy.GetParameterAsText(3)
    arcpy.AddMessage("The name of the field 2 to be combined is " + Field2 + "\n")

    # Request user input of data type = String and direction = Input
    NewField = arcpy.GetParameterAsText(4)
    arcpy.AddMessage("The name of the field to be added is " + NewField + "\n")

    # Replicate the input shapefile and add a new field to the replica
    arcpy.Copy_management(InputShapefile, OutputShapefile)
    arcpy.AddField_management(OutputShapefile, NewField, "TEXT")

    # Create an enumeration of updatable records from the shapefile's attribute table
    enumerationOfRecords = arcpy.UpdateCursor(OutputShapefile)

    # Get the name of the input shapefile's shape field
    nameOfShapeField = arcpy.Describe(InputShapefile).shapeFieldName

    # Loop through that enumeration, calculating each record's convexity
    for nextRecord in enumerationOfRecords:
        # Calculate, record, and report the shape's convexity
        CombinedValue = str(nextRecord.getValue(Field1)) + str(nextRecord.getValue(Field2))
        nextRecord.setValue(NewField,CombinedValue)
        enumerationOfRecords.updateRow(nextRecord)
        arcpy.AddMessage("   Conbined Value = " + str(nextRecord.getValue(NewField)))

    # Add a blank line at the bottom of the printed list
    arcpy.AddMessage('\n')

    # Delete row and update cursor objects to avoid locking attribute table
    del nextRecord
    del enumerationOfRecords
```

```
    # Add a blank line at the bottom of the printed list
    arcpy.AddMessage('\n')

    # Delete row and update cursor objects to avoid locking attribute table
    del nextRecord
    del enumerationOfRecords
```

Code for Add Leading Zero Tool

```
"""
THIS SCRIPT TRIES ADDING LEADING ZERO FOR ID THAT LOST LEADING ZERO IN NUMERICAL FORMAT.
To create an ArcGIS Pro script tool for this script, do the following.
1   In Catalog > Toolboxes, select an existing toolbox or create a new one.
2   Right-click on the entry for this toolbox in ArcToolbox, and use New > Script to
open a dialog box.
3   In this dialog box, use Label to name the tool being created and Script File to
specify its .py file.
        LABEL/NAME          DATA TYPE           DIRECTION
        Input Table?        Table               Input
        Field?              Field               Input
        Desired Length?     Double              Input
4   To later revise any of this, right-click to the tool's name and select Properties.
"""

# Import necessary modules
import sys, os, string, math, arcpy, traceback

# Allow output file to overwrite any existing file of the same name
arcpy.env.overwriteOutput = True

try:

    # Request user input of data type = Shapefile and direction = Input
    InputTable = arcpy.GetParameterAsText(0)
    arcpy.AddMessage('\n' + "The input shapefile name is " + InputTable)

    # Request user input of data type = String and direction = Input
    Field = arcpy.GetParameterAsText(1)
    arcpy.AddMessage("The name of the field to be added is " + Field + "\n")

    # Request user input of data type = Shapefile and direction = Input
    Length = arcpy.GetParameterAsText(2)
    arcpy.AddMessage("The desired length is " + Length)

    # Loop through the records and update fill the leading zero for the row
    # of which the length less than the desired length
    Length = int(Length)
    desired_length = Length
    with arcpy.da.UpdateCursor(InputTable, Field) as cursor:
        for row in cursor:
            row[0] = str(row[0])
            if row[0].isdigit():
                if len(row[0]) < desired_length:

                    row[0] = row[0].zfill(desired_length)
                    cursor.updateRow(row)

                elif len(row[0]) > desired_length:
                    print ("> Lenght - Manually Check: " + row[0])
            else:
                print ("NAN - Manually Check: " + row[0])

    print ("Finished")
```

```
except Exception as e:
    # If unsuccessful, end gracefully by indicating why
    arcpy.AddError('\n' + "Script failed because: \t\t" + e.args[0] )
    # ... and where
    exceptionreport = sys.exc_info()[2]
    fullermessage   = traceback.format_tb(exceptionreport)[0]
    arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```

Code for Calculate Population Density Tool

```
"""
THIS SCRIPT AIMES TO CALCULATE THE POPULATION DENSITY FOR EACH COUNTY.
To create an ArcGIS Pro script tool for this script, do the following.
1   In Catalog > Toolboxes, select an existing toolbox or create a new one.
2   Right-click on the entry for this toolbox in ArcToolbox, and use New >
Script to open a dialog box.
3   In this dialog box, use Label to name the tool being created and Script
File to specify its .py file.
        LABEL/NAME          DATA TYPE           DIRECTION
        Input Shapefile?    Feature Class       Input
        Output Shapefile?   Feature Class       Output
        Population Field    Field               Input
        Output Field?       Field               Input
4   To later revise any of this, right-click to the tool's name and select
Properties.
"""

# Import necessary modules
import sys, os, string, math, arcpy, traceback

# Allow output file to overwrite any existing file of the same name
arcpy.env.overwriteOutput = True

try:

    # Request user input of data type = Shapefile and direction = Input
    InputShapefile = arcpy.GetParameterAsText(0)
    arcpy.AddMessage('\n' + "The input shapefile name is " + InputShapefile)

    # Request user input of data type = Shapefile and direction = Output
    OutputShapefile = arcpy.GetParameterAsText(1)
    arcpy.AddMessage("The output shapefile name is " + OutputShapefile)

    # Request user input of data type = String and direction = Input
    PopulationField = arcpy.GetParameterAsText(2)
    arcpy.AddMessage("The field contain the population data is " + PopulationField + "\n")

    # Request user input of data type = String and direction = Input
    NewField = arcpy.GetParameterAsText(3)
    arcpy.AddMessage("The name of the field to be added is " + NewField + "\n")

    # Replicate the input shapefile and add a new field to the replica
    arcpy.Copy_management(InputShapefile, OutputShapefile)
    arcpy.AddField_management(OutputShapefile, NewField, "DOUBLE", 20, 5)

    # Create an enumeration of updatable records from the shapefile's attribute table
    enumerationOfRecords = arcpy.UpdateCursor(OutputShapefile)
```

```
    # Get the name of the input shapefile's shape field
    nameOfShapeField = arcpy.Describe(InputShapefile).shapeFieldName

    # Loop through that enumeration, calculating each record's convexity
    for nextRecord in enumerationOfRecords:
        # Retrieve a geometric shape from the next record's Shape field
        nextShape = nextRecord.getValue(nameOfShapeField)

        # Get that shape's area
        area = nextShape.area

        # Calculate the population density for the record
        if nextRecord.getValue(PopulationField) is None:
            continue
        else:
            PopulationDensity = float(nextRecord.getValue(PopulationField)) / area
            nextRecord.setValue(NewField,PopulationDensity)
            enumerationOfRecords.updateRow(nextRecord)
            arcpy.AddMessage("   Population Density = " + str(nextRecord.getValue(NewField)))

    # Add a blank line at the bottom of the printed list
    arcpy.AddMessage('\n')

    # Delete row and update cursor objects to avoid locking attribute table
    del nextRecord
    del enumerationOfRecords

except Exception as e:
    # If unsuccessful, end gracefully by indicating why
    arcpy.AddError('\n' + "Script failed because: \t\t" + e.args[0] )
    # ... and where
    exceptionreport = sys.exc_info()[2]
    fullermessage   = traceback.format_tb(exceptionreport)[0]
    arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```

Code for COVID Risk Classification Tool

```
"""
THIS SCRIPT AIMES TO CLASSIFY THE RISK CLASS FOR EACH COUNTY.
To create an ArcGIS Pro script tool for this script, do the following.
1   In Catalog > Toolboxes, select an existing toolbox or create a new one.
2   Right-click on the entry for this toolbox in ArcToolbox, and use New > Script
to open a dialog box.
3   In this dialog box, use Label to name the tool being created and Script File
to specify its .py file.
        LABEL/NAME              DATA TYPE           DIRECTION
        Input Shapefile?        Feature Class       Input
        Output Shapefile?       Feature Class       Output
        Covid Cases             Field               Input
        Covid Cases Speed       Field               Input
        Output Field?           Field               Input
4   To later revise any of this, right-click to the tool's name and select Properties.
"""

# Import necessary modules
import sys, os, string, math, arcpy, traceback

# Allow output file to overwrite any existing file of the same name
arcpy.env.overwriteOutput = True

try:

    # Request user input of data type = Shapefile and direction = Input
    InputShapefile = arcpy.GetParameterAsText(0)
    arcpy.AddMessage('\n' + "The input shapefile name is " + InputShapefile)
```

```
# Request user input of data type = Shapefile and direction = Output
OutputShapefile = arcpy.GetParameterAsText(1)
arcpy.AddMessage("The output shapefile name is " + OutputShapefile)

# Request user input of data type = String and direction = Input
CasesField = arcpy.GetParameterAsText(2)
arcpy.AddMessage("The field contain the confirmed Covid-19 cases data is " +
                 CasesField + "\n")

# Request user input of data type = String and direction = Input
SpeedField = arcpy.GetParameterAsText(3)
arcpy.AddMessage("The field contain the confirmed Covid-19 cases change speed data is "
                 + SpeedField + "\n")

# Request user input of data type = String and direction = Input
NewField = arcpy.GetParameterAsText(4)
arcpy.AddMessage("The name of the field to be added is " + NewField + "\n")


# Create an enumeration of updatable records from the shapefile's attribute table
enumerationOfRecords = arcpy.UpdateCursor(OutputShapefile)

# Get the name of the input shapefile's shape field
nameOfShapeField = arcpy.Describe(InputShapefile).shapeFieldName

# Loop through that enumeration, calculating each record's convexity
for nextRecord in enumerationOfRecords:
    # Calculate the population density for the record
    if nextRecord.getValue(CasesField) is None:
        continue

    if nextRecord.getValue(SpeedField) is None:
        continue

    else:
        if float(nextRecord.getValue(CasesField)) >= 10000 and
        float(nextRecord.getValue(SpeedField)) > 150:
            RiskClass = 5
            nextRecord.setValue(NewField,RiskClass)
            enumerationOfRecords.updateRow(nextRecord)
            arcpy.AddMessage("  Covid-19 Risk Class = " + str(nextRecord.getValue(NewField)))
        if float(nextRecord.getValue(CasesField)) >= 10000 and
        float(nextRecord.getValue(SpeedField)) <= 150:
            RiskClass = 4
            nextRecord.setValue(NewField,RiskClass)
            enumerationOfRecords.updateRow(nextRecord)
            arcpy.AddMessage("  Covid-19 Risk Class = " + str(nextRecord.getValue(NewField)))
        if 4000 <= float(nextRecord.getValue(CasesField)) <10000:
            RiskClass = 4
            nextRecord.setValue(NewField,RiskClass)
            enumerationOfRecords.updateRow(nextRecord)
            arcpy.AddMessage("  Covid-19 Risk Class = " + str(nextRecord.getValue(NewField)))
        if 1000 <= float(nextRecord.getValue(CasesField)) <4000:
            RiskClass = 3
            nextRecord.setValue(NewField,RiskClass)
            enumerationOfRecords.updateRow(nextRecord)
            arcpy.AddMessage("  Covid-19 Risk Class = " + str(nextRecord.getValue(NewField)))
        if float(nextRecord.getValue(CasesField)) <1000 and
        float(nextRecord.getValue(SpeedField)) >= 50:
            RiskClass = 2
            nextRecord.setValue(NewField,RiskClass)
            enumerationOfRecords.updateRow(nextRecord)
            arcpy.AddMessage("  Covid-19 Risk Class = " + str(nextRecord.getValue(NewField)))
        if float(nextRecord.getValue(CasesField)) <1000 and
        float(nextRecord.getValue(SpeedField)) < 50:
            RiskClass = 1
            nextRecord.setValue(NewField,RiskClass)
            enumerationOfRecords.updateRow(nextRecord)
            arcpy.AddMessage("  Covid-19 Risk Class = " + str(nextRecord.getValue(NewField)))

# Add a blank line at the bottom of the printed list
arcpy.AddMessage('\n')
```

```
    # Delete row and update cursor objects to avoid locking attribute table
    del nextRecord
    del enumerationOfRecords

except Exception as e:
    # If unsuccessful, end gracefully by indicating why
    arcpy.AddError('\n' + "Script failed because: \t\t" + e.args[0] )
    # ... and where
    exceptionreport = sys.exc_info()[2]
    fullermessage   = traceback.format_tb(exceptionreport)[0]
    arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```

Googel Earth Engine Code link

https://code.earthengine.google.com/171bd2cbf95a0ce97a2ec9af79208bc0