Model Methodology

1. Data Processing

a. Data Preparation

Box link that contains all the data we need: https://uofi.app.box.com/folder/80580683506

Data transformation:

1) Calculate the state-level average case numbers for each county-year pair.

3) Calculate average cases of neighbor counties for each county-year pair. Neighbor information are found on the Internet.

4) Download and include each county's county type.

5) Download weather data, include air temperature, humidity and precipitation. Also calculate the rainfall gini index.

b. Required process by CDC manual

1) Change fips '51019/51515' to 51019

3) Change fips '46102/46113' to 46102

4) Other changes to the name of counties of locations.

c. Create lagged variables

Our regression formula is:

count (at time t) ~ count (at time t-2) + count (at time t-3) + count (at time t-4) + count (at time t-5) +count (at time t-6) + other variables(at time t-2)

And we just create corresponding lagged variables.

Note that:

1) The most recent data is **2 years prior** than the target year, because we only have data until 2018, and are going to predict 2020 results. We don't have data for 2019, so we can't use 1-year lags.

2) Because we need count data that is 6-year prior than the target year as variables, and for year 2000-2005 we don't have complete past records, they cannot be the target. So our **final dataframe only contains data from 2006-2018**.

d. Data discription

1) Data shape

Our final dataset contains 40391 rows, which can be calculated by:

3108 counties * 13 years - some bad records = 40391

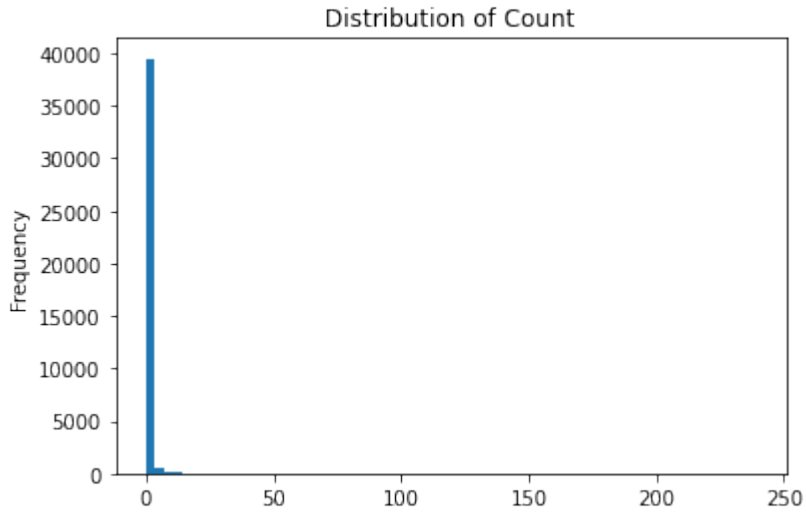
The columns are:

0	GEOID	int64	The fips of county
1	year	int64	Target year t (2006-2018)
2	count	int64	Neuro-invasive case count
3	count_lag2	int64	Count data of t - 2
4	count_lag3	int64	Count data of t - 3
5	count_lag4	int64	Count data of t - 4
6	count_lag5	int64	Count data of t - 5
7	count_lag6	int64	Count data of t - 6
8	stateAvg_lag2	float64	Average neuro-invasive cases in its state at t - 2
9	Incident_nominal_lag2	int64	The official incident cases at t - 2. If t - 2 is prior than the incident year, it equals to 0
10	neigh_death_count_lag2	int64	Sum of death cases in neibor counties at t - 2
11	temp_lag2	float64	Average air temperature at surface of year t - 2
12	prec_lag2	float64	Average precipitation of year t - 2
13	hum_lag2	float64	Average humility of year t - 2
14	gini_lag2	float64	Precipitation gini index of year t - 2
15	County_type_lag2	int64	County type at t - 2. Barely changes so it has no difference for any year.

2) Count distribution

From the chart and table, we can see that the **count data has more than 75% 0s**, and has a few extreme values(like 239). Intuitively, it would be easier to predict 0 than predict larger values, because the model would be "overwhelmed" by 0 records. And we might have to treat zero and nonzero cases with seperate models.

Also, **the variance(std^2) of count is way larger than the mean**. In this case, a negative binomial model(2 scale parameters) would be better than a poisson model(1 scale parameter and mean = variance). We will talk about it more in part 2.

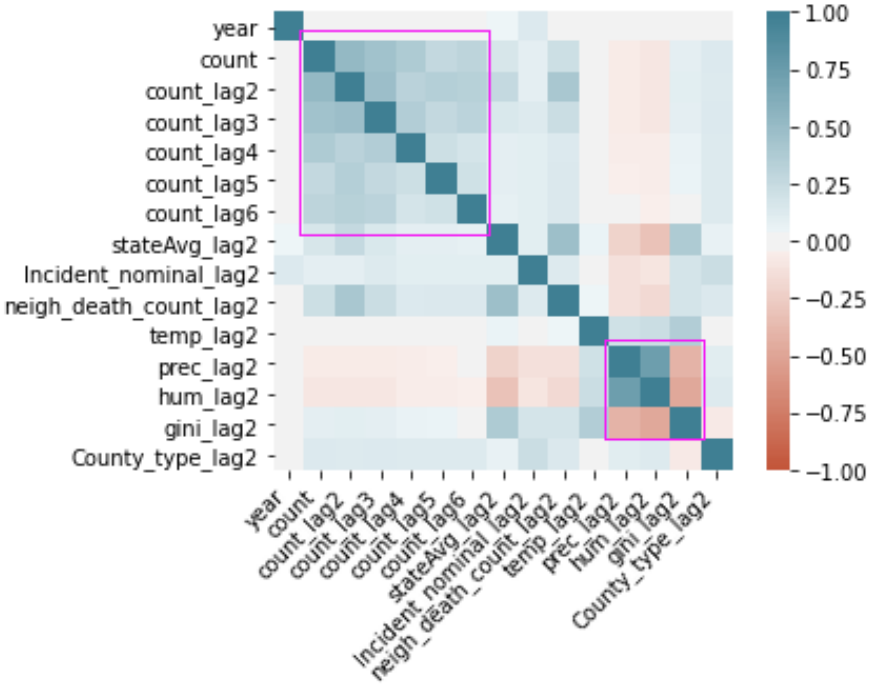



```
print(data_lagged['count'].describe())
```

```
count    40391.000000
mean         0.401946
std          3.451856
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max        239.000000
```

3) Correlation between predictors

Here I plot the correlation between columns except for 'GEOID'. Look at the left upper block and right bottom block. There is some relationship between count and its lags, and the correlation decreases when we track to earlier years. And there is a strong correlation among precipitation/humility/precipitation gini. (Though it's strange why humility and precipitation are negatively correlated) This makes sense because they are all related to precipitation in an area. And it tells us that we don't have to include all 3 of them in our modeling.



2. Modeling

a. Ordinary least square regression

1) Regression formula

Dependent variable: count

Independent variables: year + count_lag2 + count_lag3 + count_lag4 + count_lag5 + count_lag6 + stateAvg_lag2 + Incident_nominal_lag2 + neigh_death_count_lag2 + temp_lag2 + prec_lag2 + hum_lag2 + gini_lag2 + County_type_lag2

(Weather data is included here. Column types could be found above.)

Transformation: None

Samples: All 40391 observations read from jing_lagged_data.csv. We don't split out a validation set here, because we are only interested in model diagnostics especially finding outliers, not the performance of linear regression.

2) Regression summary

The R-square is 0.371, which means OLS is not fitting very well on our data. Also, the skew of the data is as large as 30, and a high skew of data impairs the normality assumption of OLS. We can also get it from the above distribution histogram.

OLS Regression Results

Dep. Variable:	count	R-squared:	0.371
Model:	OLS	Adj. R-squared:	0.371
Method:	Least Squares	F-statistic:	1703.
Date:	Fri, 24 Jul 2020	Prob (F-statistic):	0.00
Time:	12:03:06	Log-Likelihood:	-97983.
No. Observations:	40391	AIC:	1.960e+05
Df Residuals:	40376	BIC:	1.961e+05
Df Model:	14		
Covariance Type:	nonrobust		

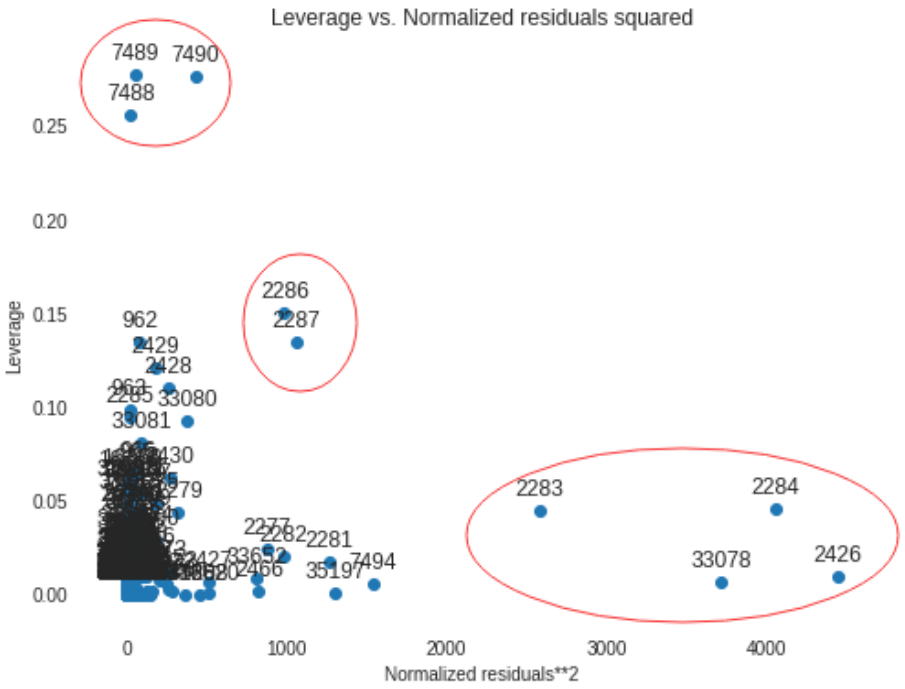
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-41.8858	7.442	-5.629	0.000	-56.472	-27.300
year	0.0207	0.004	5.604	0.000	0.013	0.028
count_lag2	0.3248	0.005	63.878	0.000	0.315	0.335
count_lag3	0.1780	0.005	38.675	0.000	0.169	0.187
count_lag4	0.1481	0.004	41.783	0.000	0.141	0.155
count_lag5	0.0430	0.004	11.437	0.000	0.036	0.050
count_lag6	0.0783	0.004	20.588	0.000	0.071	0.086
stateAvg_lag2	0.0522	0.019	2.805	0.005	0.016	0.089
Incident_nominal_lag2	-0.0320	0.029	-1.094	0.274	-0.089	0.025
neigh_death_count_lag2	-0.0063	0.002	-3.850	0.000	-0.010	-0.003
temp_lag2	0.0006	0.001	0.482	0.630	-0.002	0.003
prec_lag2	-0.0517	0.022	-2.362	0.018	-0.095	-0.009
hum_lag2	-0.0021	0.002	-0.948	0.343	-0.006	0.002
gini_lag2	0.6059	0.232	2.609	0.009	0.151	1.061
County_type_lag2	0.2818	0.033	8.624	0.000	0.218	0.346

Omnibus:	105059.332	Durbin-Watson:	1.527
Prob(Omnibus):	0.000	Jarque-Bera (JB):	5004900536.738
Skew:	29.893	Prob(JB):	0.00
Kurtosis:	1726.455	Cond. No.	1.11e+06

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.11e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

3) Model diagnostics

This is a residuals vs. leverage plot of ordinary least square regression, which help us to identify influential points and outliers. Residuals are the difference between actual y and regressed or predicted y. Leverages measure the influence of one point to the regression result, and is only determined by the independet variables. Typically if a point has a high leverage, it has larger influence on the regression, and the residual would be relatively smaller. The numbers on the plot are the indices of points, so that we can locate them. We can see that there are three kinds of outliers: first is the bottom right cluster where points have high residuals but low influences; second the upper left cluster where points have higher leverage and low residuals; and the last one is 2286/2287, who have both large residuals and high leverage.



Let's first take a look at these two clusters. First is the bottom right one with large residuals and low leverages. The residuals are large because they have high actual counts but the predicted count is much lower. And their leverages are low, due to that their independent variables especially prior counts are not that abnormal. This situation is similar to a suddenly outbreak of pandemic.

	GEOID	year	count	predict_y	count_lag2	count_lag3	count_lag4	count_lag5	count_lag6
2283	6037	2014	195	55.902567	119	41	3	15	107
2284	6037	2015	239	64.613430	109	119	41	3	15
2426	6059	2014	197	14.556010	26	10	1	2	54
33078	48113	2012	175	8.232359	0	13	3	12	59

The middle cluster contains two observations: 2286 and 2287. They have relatively large residuals and large leverages. Generally we will say that these two points are outliers in OLS. Also notice that many of the points in the bottom right cluster and middle cluster belong to GEOID 06037, which is the Los Angelas City. We should pay extra attention to it because it has large counts as long as a very high variance. Compared to records of other years, these two records have much more historical cases in 5 prior years. But their actual count values in current year are distinguished, one pretty higher and one pretty lower. What possibly happened is that in 2017 LA took control to the virus, so the pandemic went insignificant in 2018.

	GEOID	year	count	predict_y	count_lag2	count_lag3	count_lag4	count_lag5	count_lag6
2286	6037	2017	223	137.156851	239	195	109	119	41
2287	6037	2018	34	123.141388	114	239	195	109	119

Finally let's look at the upper left cluster. They are more easy to interprete. These three records belong to the Cook county in IL. They have low residuals, though the residual of index 7490 seems a bit large. And they have abnormal historical records. The record with 398 cases belongs to year 2002. It's not shown in this table because we didn't do prediction before 2006. After searching from Internet we get some relavant information: in 2002, Illinois experienced a severe WNV out break, and Cook county was the most influenced one[1][2]. However, the pandemic was controled in the next year, so the records in later years went lower and lower.

	GEOID	year	count	predict_y	count_lag2	count_lag3	count_lag4	count_lag5	count_lag6
7488	17031	2006	53	66.109903	13	16	398	0	0
7489	17031	2007	26	45.397816	73	13	16	398	0
7490	17031	2008	7	63.791270	53	73	13	16	398

b. Point estimation: a two-stage model

1) Intuition

There are too many zero counts in the data, which we are not that interested in and we don't want them to overwhelm our model. So we decide to fit a two-stage model, which is a zero/nonzero classification model followed by a regression model. At the first stage, we wipe out some of the trivial records and give them zero predictions, so that the following regression model could be less influenced by those zero counts. Then at the second stage, we run a regression model to estimate the excact count for the rest of the data.

The actual procedure is: first we fit a random forest classification on the data, get a prediction of zero or nonzero. Then for the predicted nonzeros, we perform another OLS regression for their point estimation, and we coerce the predictions to be nonzero integers. The final result is either 0(given by random forest) or y(given by linear regression).

2) Models

Model 1 - Random forest with for zero/nonzero classification

Parameters: n_estimators=120, others default as sklearn.ensemble.RandomForestClassifier.

Model 2 - Ordinary least square regression

Dependent variables: count

Independent variables: year + count_lag2 + count_lag3 + count_lag4 + count_lag5 + count_lag6 + stateAvg_lag2 + Incident_nominal_lag2 + neigh_death_count_lag2 + temp_lag2 + prec_lag2 + hum_lag2 + gini_lag2 + County_type_lag2

c. Bin classification: simple random forest

1) Intuition

We use a naive random forest model to do the prediction of bin probabilities. The probabilities are calculated based on the voting results of trees in the random forest. In a basic example, if the random forest is composed of 10 trees, and 9 of them give a prediction of 0, while the left 1 tree give a prediction of 2, then the probability for 0 is 90% and probability for 2 is 10%, leaving all other bins with 0 probability.

2) Model: Random forest classification

Parameters: n_estimators=300, others default as sklearn.ensemble.RandomForestClassifier.

1. https://www.dph.illinois.gov/topics-services/diseases-and-conditions/west-nile-virus
2. https://www.nsmad.com/about-mosquitoes/west-nile-virus/