Machine Learning on Covid-19

Presented by: Jiarong Chen

Background

In 2020 and the future, covid-19 will be a hot topic and a big challenge for our human. For my health and life, I am tracking the covid status all the times. For the machine learning practice, I am going to make a project related to Covid-19. In this project, I'll be working with the Covid-19 dataset until 2020/11/10 from https://covidtracking.com/data/download.

01.

The Dangerous States

Death Rate and Positive Rate

02.

The Status of New York

Death cases increasing daily and Positive cases increasing daily

03.

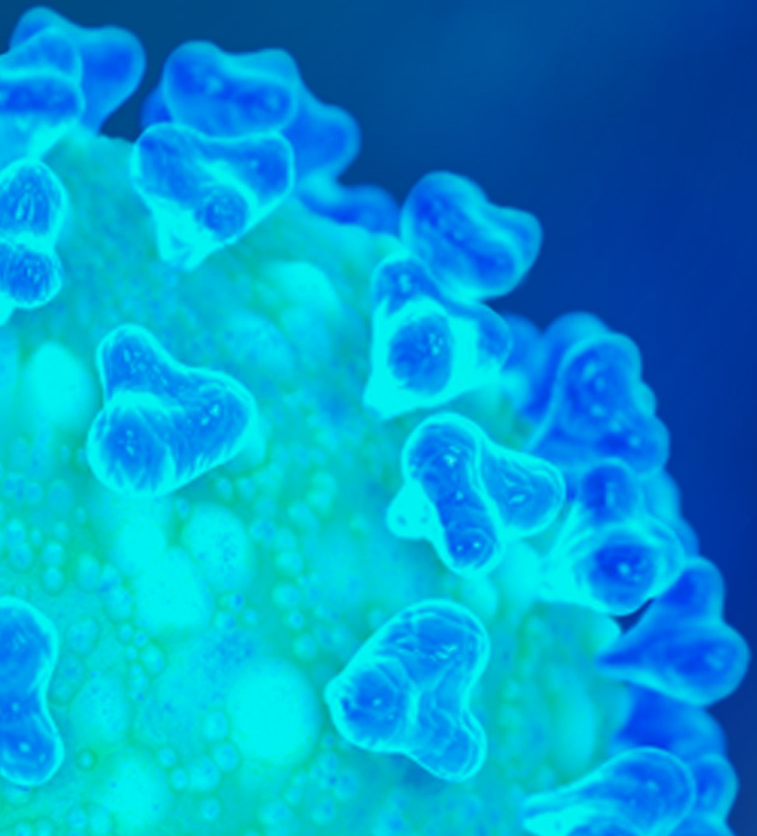
Classification Model

Logistic Regression, Random Forest, XGBoost

04.

Summary

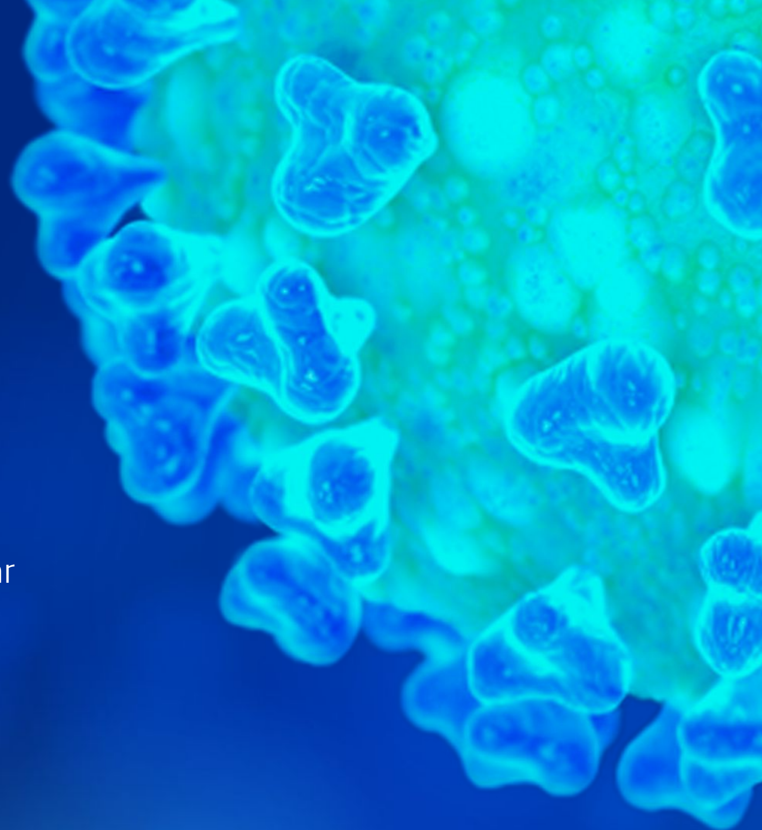
Conclusion, Recommendation, Future work



COVID-19

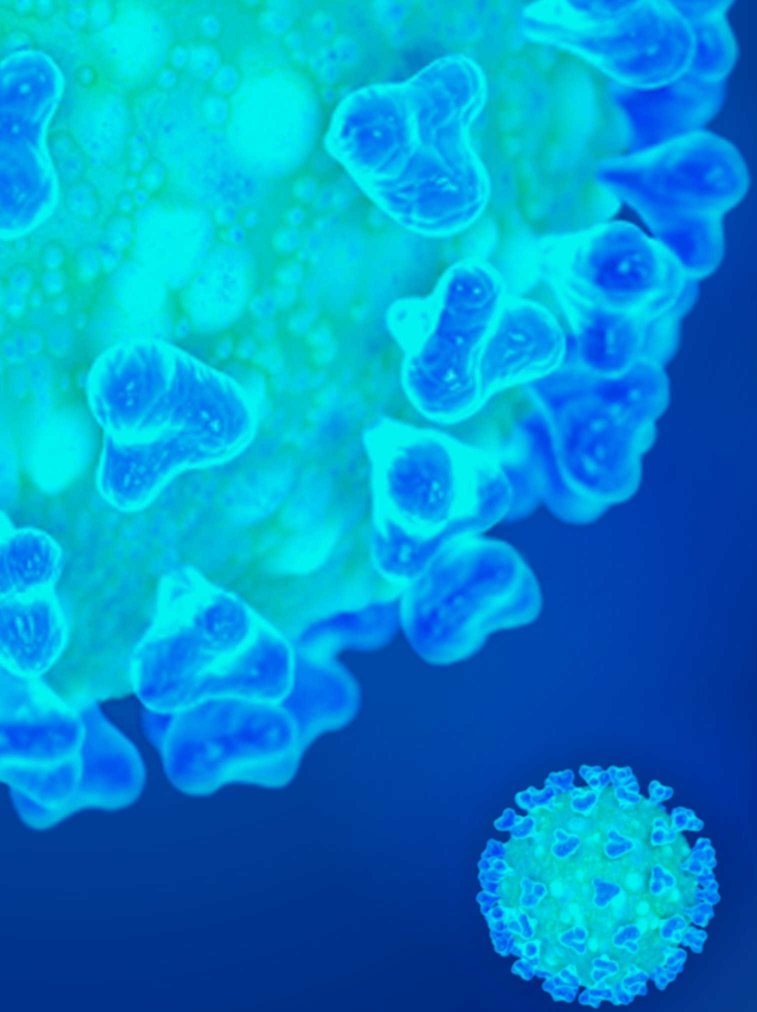
COVID-19 is an infectious disease caused by the recently found virus known as SARS-CoV-2 (or coronavirus). Before the outbreak originated in Wuhan, China on December 2019, there was no information about this virus

HISTORY



The oldest common ancestor of coronavirus has been dated as far back as the 9th century BC. Some studies published in 1990 specified the most recent common ancestors as follows:

- Betacoronavirus: 3300 BC
- Deltacoronavirus: 3000 BC
- Gammacoronavirus: 2800 BC
- Alphacoronavirus: 2400 BC



01.

The Dangerous States

By Death Cases Ranking

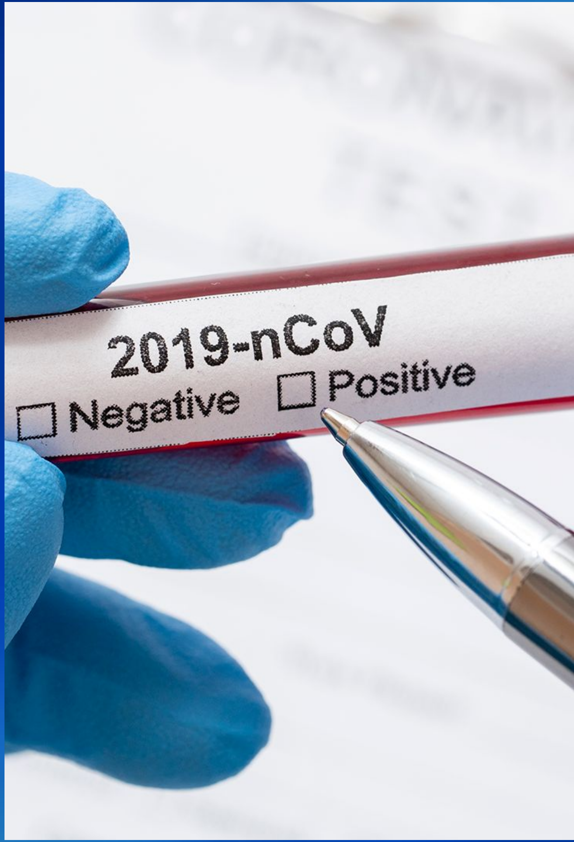
Death Rate

	deathRate	death
state		
NJ	0.00185089	16440.0
MA	0.00146241	10163.0
NY	0.00133513	25973.0
CT	0.00131771	4698.0
LA	0.00130098	6048.0
RI	0.00116391	1233.0
MS	0.00115686	3443.0
DE	0.00101878	719.0
AZ	0.000846853	6164.0
MI	0.000801854	8008.0

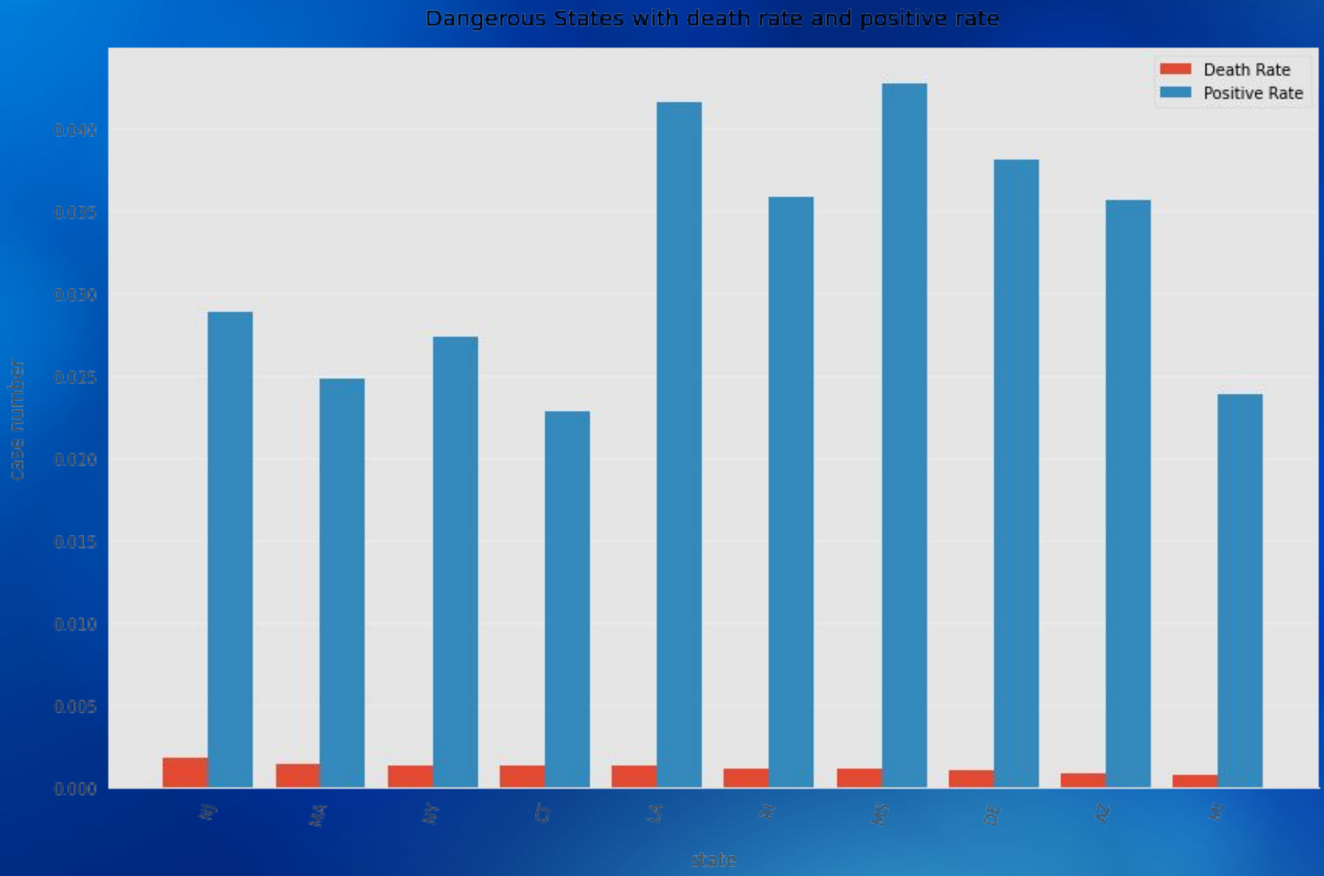
By Positive Cases Ranking

Positive Rate

state	positiveRate	positive
ND	0.0727736	55458.0
SD	0.0636528	56311.0
WI	0.0491856	286380.0
IA	0.0459331	144922.0
NE	0.0434081	83969.0
MS	0.0427415	127205.0
TN	0.0421137	287770.0
UT	0.0420679	134868.0
AL	0.0417804	204857.0
LA	0.0415962	193372.0



Dangerous states with death rate and positive rate



Highest Rate Report:

0.185%

Highest Death Rate from NJ

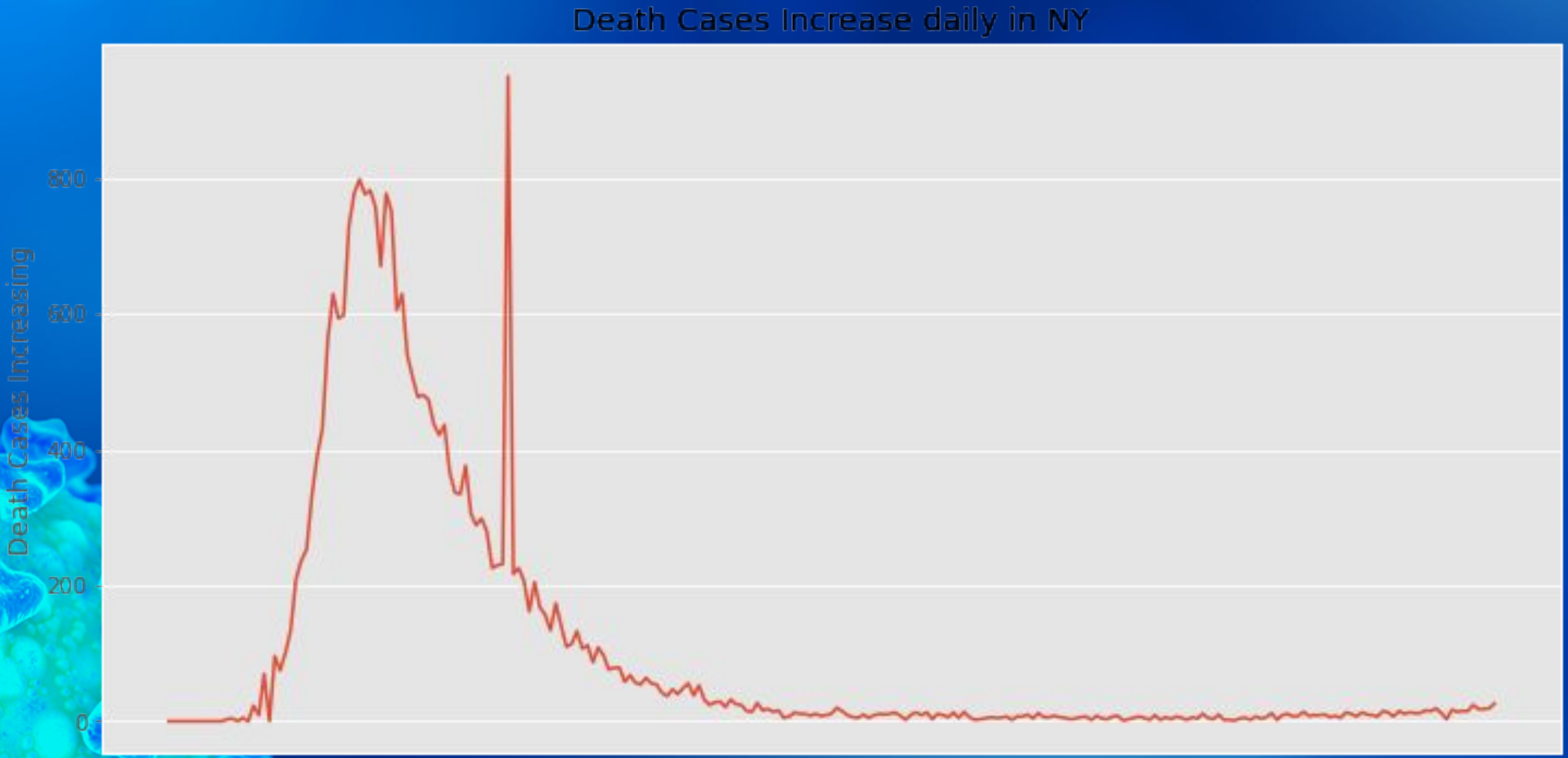
7.28%

Highest Positive Rate from ND

02

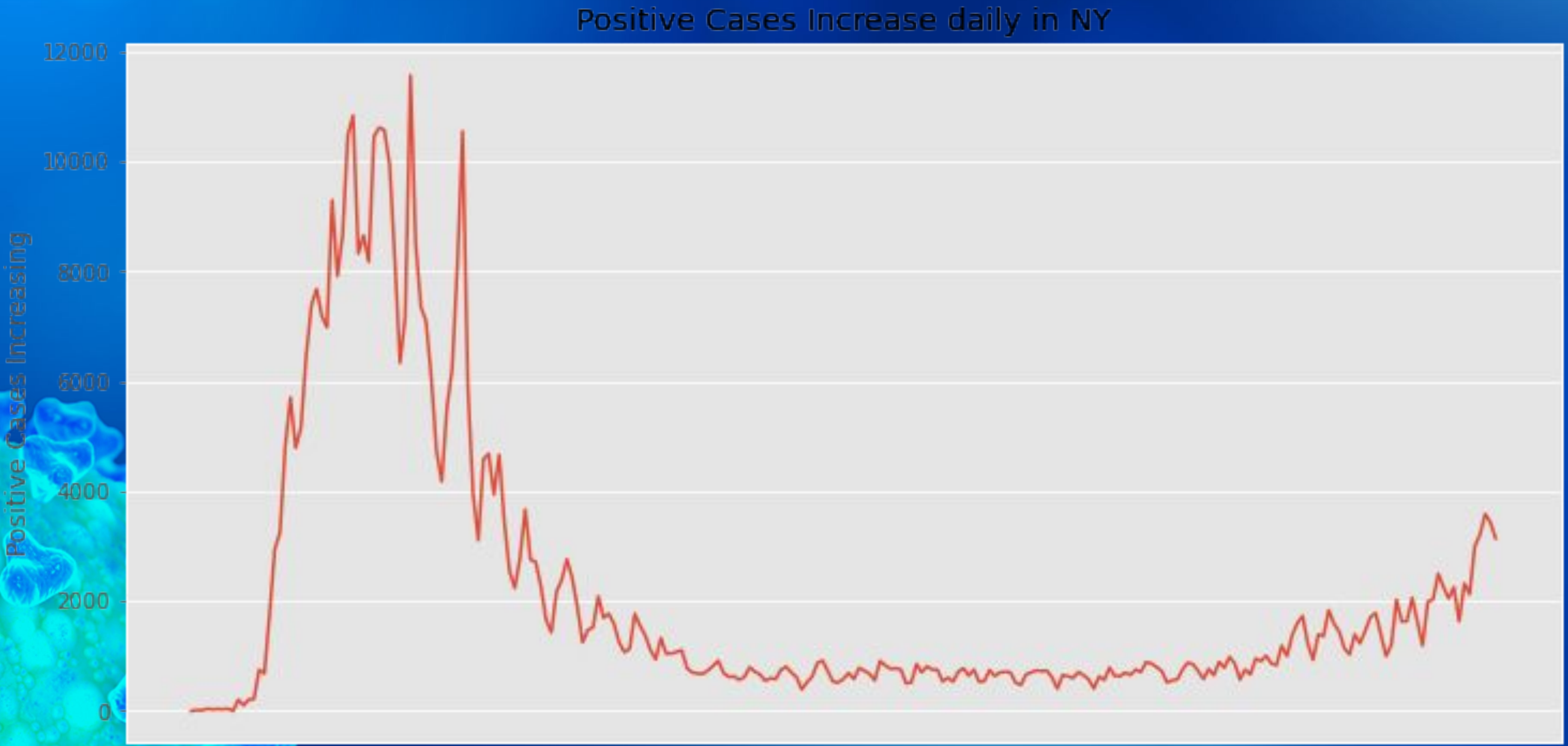
The Status of New York State

Death Cases



Death Cases Increase daily in NY
Death Cases Increasing
800
600
400
200
0

Positive Cases



Stable?

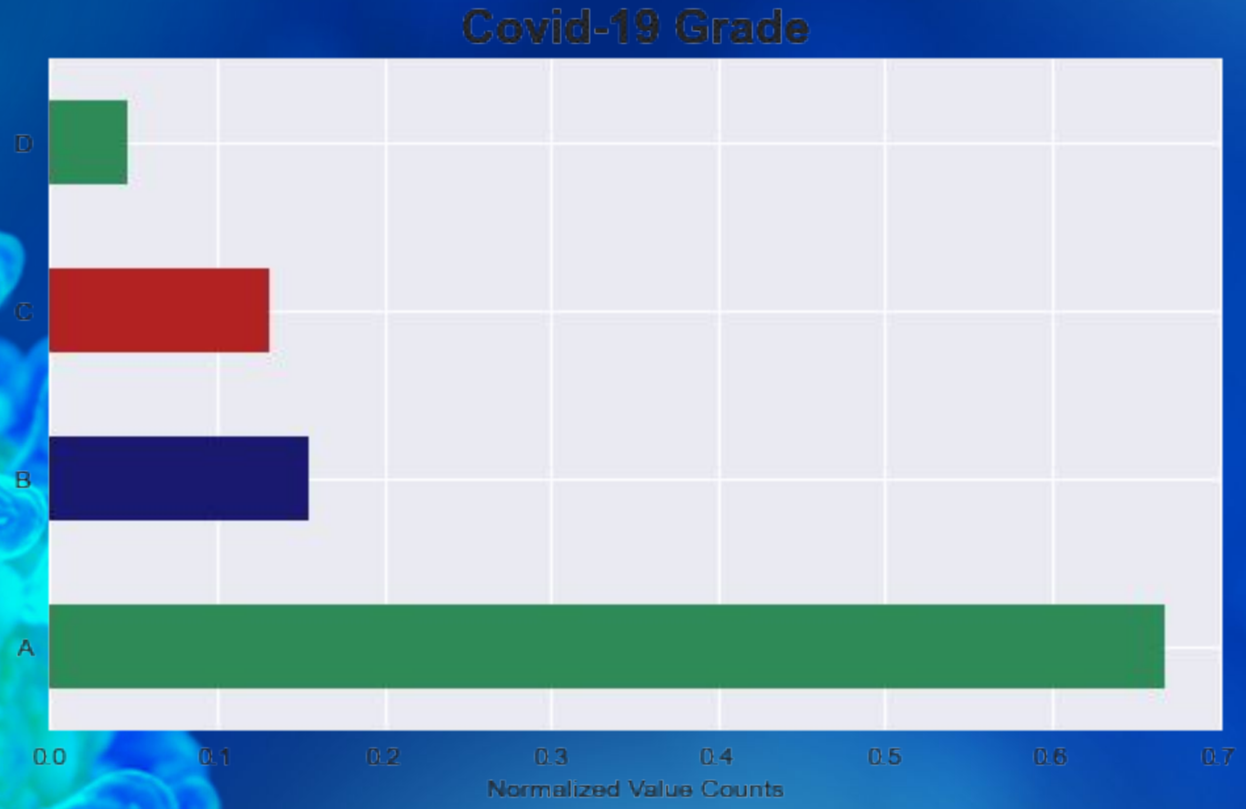
Rebound?

03

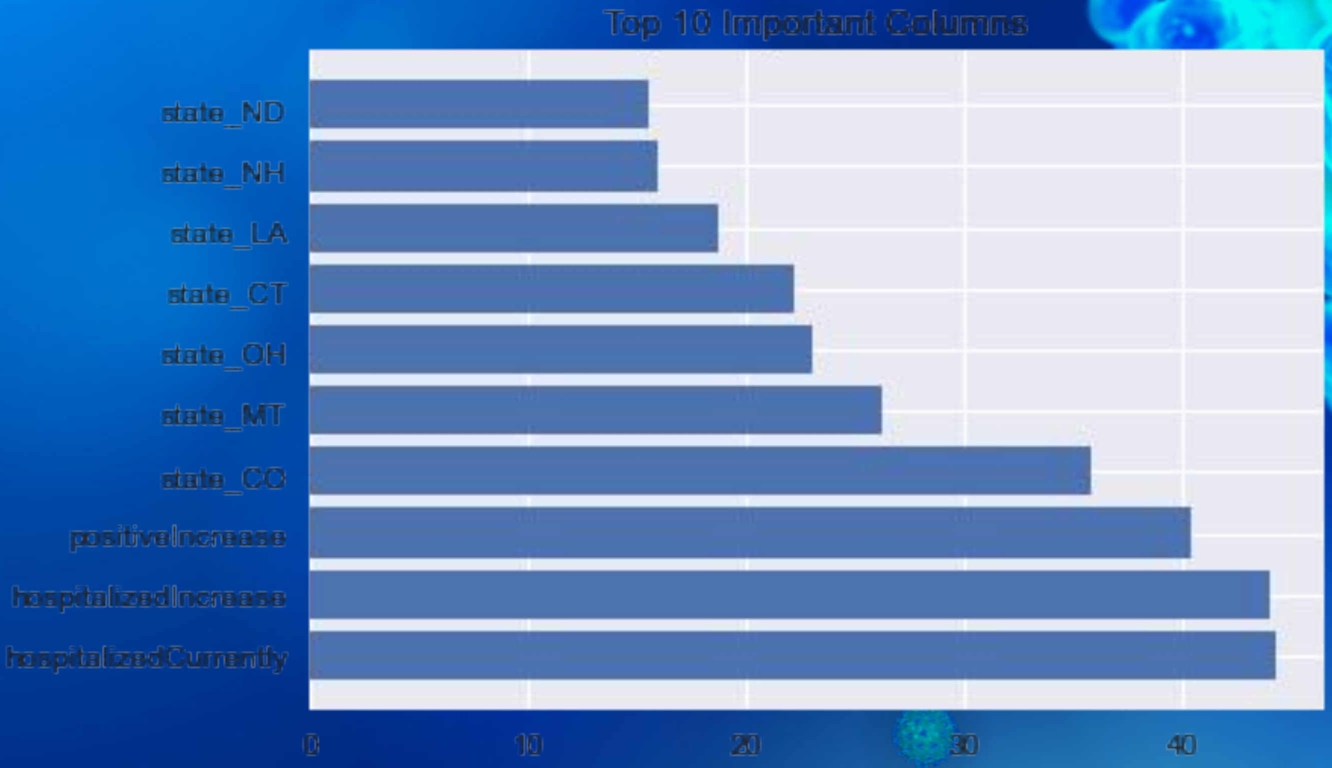
Classification model

To predict the covid-19 quality daily

Covid-19 Grade



Top 10 importance features



Model Information

Cost Benefit: -18.85

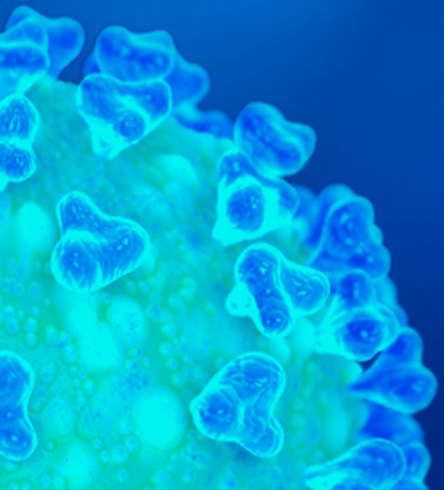
Confusion Matrix:

```
[[785 112  50   7]
 [ 29  28  20   3]
 [ 83  86 115  26]
 [  8   7  13  41]]
```

Training Accuracy: 71.86%

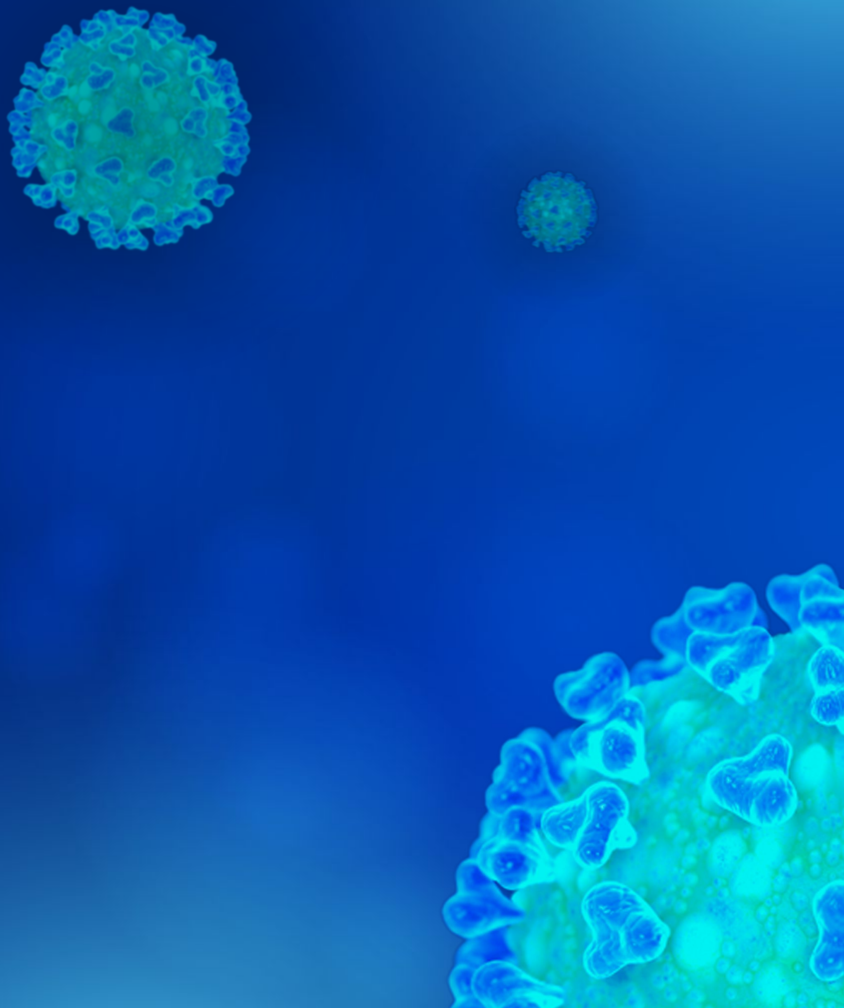
Test Accuracy: 68.58%

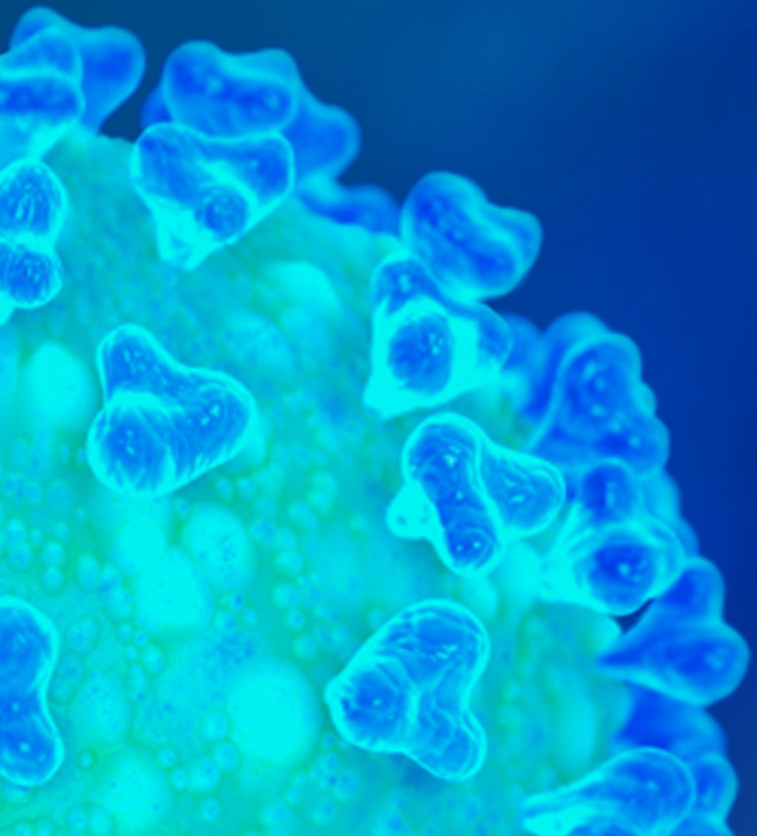
Validation Accuracy: 72.23%



04

Summary





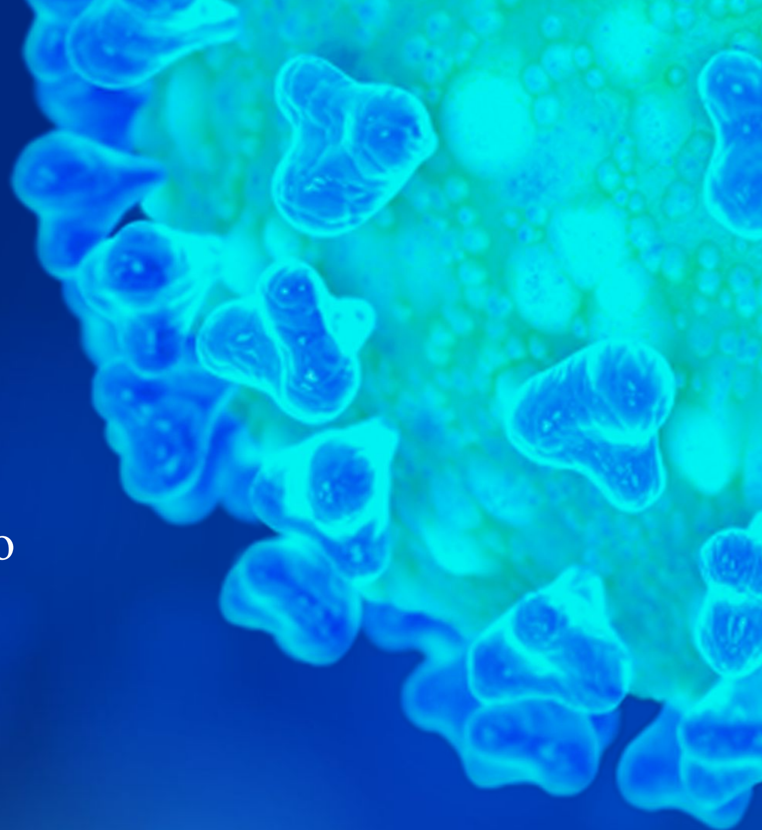
Conclusion & Recommendation

For residents: Even though covid-19 of some states are stable, we still can not relax our guard. Covid-19 is a serious problem in our life, and spreads very quickly. In some dangerous states, such as New York state, there is a rick of rebound. What we can do now are quarantining and keeping social distance. Do not travel around, especially some dangerous states.

For analysts: GridsearchCV is a very useful tool to help us tune model parameters.

Future Work

Accuracy: Find a better or some better parameters to improve accuracy
Overfitting: Deal with the overfitting of random forest model and XGBoost
More analyses: Try analyse other columns of the dataset, such as recover
Death Rate: The death rate of New York state is high. Try to find some reasons and related data

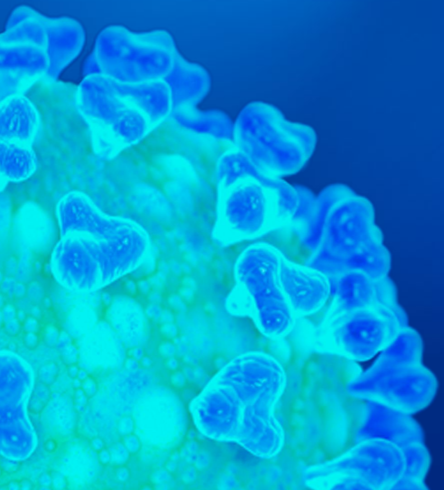


THANKS!

Any question?
Find me at:
jiarong.chen.1@alumni.stonybrook.edu
https://www.linkedin.com/in/jiarong-jr-chen-ba87b214a/
https://github.com/JRRRRRRR

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik

Appendix



Logistic Regression

Logistic Regression

```python
# Instantiate the LogisticRegression for GridsearchCV
logre = LogisticRegression(penalty='l1', tol=0.01)
```

```python
# Parameter Tuning
# Create a Grid for GridsearchCV to find the best parameters for the classifier
rf_param_grid = {'C':[1, 0.1, 1.0],  'class_weight':[None, 'balanced'], 'solver':['saga','newton-cg']}

gs_tree = GridSearchCV(logre, rf_param_grid, cv=3)

gs_tree.fit(X_validation, y_validation)
print(gs_tree.best_params_)
```

```
{'C': 1, 'class_weight': None, 'solver': 'saga'}
```

```python
# Apply the best_params to the model
logre = LogisticRegression(C=1, penalty='l1', tol=0.01, solver='saga')
logre.fit(X_train, y_train)
run_model(logre, X_train, X_test, y_train, y_test, X_validation, y_validation)
```

```
Training Accuracy:   71.86%
Test Accuracy:       68.44%
Validation Accuracy: 72.23%
```

Random Forest

Random Forest

```python
# Instantiate the RandomForestClassifier for GridsearchCV
forest = RandomForestClassifier()
```

```python
# Parameter Tuning
# Create a Grid for GridsearchCV to find the best parameters for the classifier
rf_param_grid = {'n_estimators': [20, 50, 100,],
                 'criterion': ['gini', 'mse'],
                 'max_depth': [None, 2, 3, 5, 7],
                 'min_samples_split': [2, 5, 8],
                 'min_samples_leaf': [2, 5, 8],
                 'max_features':np.arange(5,10)}

gs_tree = GridSearchCV(forest, rf_param_grid, cv=3)

gs_tree.fit(X_validation, y_validation)
print(gs_tree.best_params_)
```

```
{'criterion': 'gini', 'max_depth': None, 'max_features': 6, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimat
ors': 100}
```

```python
# Apply the best_params to the model
forest = RandomForestClassifier(n_estimators=100, criterion='gini',
                                max_depth=None, min_samples_split=2,
                                min_samples_leaf=2, max_features=6, random_state=17)
forest.fit(X_train, y_train)
run_model(forest, X_train, X_test, y_train, y_test, X_validation, y_validation)
```

```
Training Accuracy:   93.22%
Test Accuracy:       72.26%
Validation Accuracy: 70.19%
```

XGBoost

XGBoost

```
# Instantiate the XGBoost Classifier for GridsearchCV
xgboost = xgb.XGBClassifier()
```

```
# Parameter Tuning
# Create a Grid for GridsearchCV to find the best parameters for the classifier
xg_param_grid = {'n_estimators': [20, 30, 100, 250],
                 'learning_rate': [0.05, 0.1, 0.15],
                 'max_depth': [2, 3, 5, 7],
                 'colsample_bytree': [0.7, 1],
                 'gamma': [0.0, 0.1]}

xg_search = GridSearchCV(xgboost, xg_param_grid, scoring='accuracy', cv=3)

xg_search.fit(X_validation, y_validation)

print(xg_search.best_params_)
```

```
{'colsample_bytree': 0.7, 'gamma': 0.0, 'learning_rate': 0.15, 'max_depth': 7, 'n_estimators': 30}
```

```
# Apply the best_params to the model
booster = xgb.XGBClassifier(learning_rate=0.15, max_depth=7, n_estimators=30,
                            colsample_bytree=0.7, gamma=0.0, random_state=17)
booster.fit(X_train, y_train)
run_model(booster, X_train, X_test, y_train, y_test, X_validation, y_validation)
```

```
Training Accuracy:   85.08%
Test Accuracy:       69.92%
Validation Accuracy: 70.19%
```

Logistic Regression(SMOTE)

Logistic Regression (SMOTE):

```
# Parameter Tuning
# Create a Grid for GridsearchCV to find the best parameters for the classifier
rf_param_grid = {'C':[1, 0.1, 1.0],  'class_weight':[None, 'balanced'], 'solver':['saga','newton-cg']}

gs_tree = GridSearchCV(logre, rf_param_grid, cv=3)

gs_tree.fit(X_validation2, y_validation2)
print(gs_tree.best_params_)
```

```
{'C': 1, 'class_weight': None, 'solver': 'saga'}
```

```
# Apply the best_params to the model
logre2 = LogisticRegression(C=1, penalty='l1', tol=0.01, solver='saga')
logre2.fit(X_train2, y_train2)
run_model(logre2, X_train2, X_test2, y_train2, y_test2, X_validation2, y_validation2)
```

```
Training Accuracy:   62.59%
Test Accuracy:       57.79%
Validation Accuracy: 57.72%
```

Random Forest(SMOTE)

Random Forest (SMOTE):

```
# Parameter Tuning
# Create a Grid for GridsearchCV to find the best parameters for the classifier
rf_param_grid = {'n_estimators': [20, 50, 100,],
                 'criterion': ['gini', 'mse'],
                 'max_depth': [None, 2, 3, 5, 7],
                 'min_samples_split': [2, 5, 8],
                 'min_samples_leaf': [2, 5, 8],
                 'max_features':np.arange(5,10)}

gs_tree = GridSearchCV(forest, rf_param_grid, cv=3)

gs_tree.fit(X_validation2, y_validation2)
print(gs_tree.best_params_)
```

```
{'criterion': 'gini', 'max_depth': None, 'max_features': 9, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimat
ors': 50}
```

```
# Apply the best_params to the model
forest2 = RandomForestClassifier(n_estimators=50, criterion='gini',
                                 max_depth=None, min_samples_split=5,
                                 min_samples_leaf=2, max_features=9, random_state=17)
forest2.fit(X_train2, y_train2)
run_model(forest2, X_train2, X_test2, y_train2, y_test2, X_validation2, y_validation2)
```

```
Training Accuracy:   96.05%
Test Accuracy:       73.44%
Validation Accuracy: 71.31%
```

XGBoost(SMOTE)

XGBoost (SMOTE):

```
# Parameter Tuning
# Create a Grid for GridsearchCV to find the best parameters for the classifier
xg_param_grid = {'n_estimators': [20, 30, 100, 250],
                 'learning_rate': [0.05, 0.1, 0.15],
                 'max_depth': [2, 3, 5, 7],
                 'colsample_bytree': [0.7, 1],
                 'gamma': [0.0, 0.1]}

xg_search = GridSearchCV(xgboost, xg_param_grid, scoring='accuracy', cv=6)

xg_search.fit(X_validation2, y_validation2)

print(xg_search.best_params_)
```

```
{'colsample_bytree': 1, 'gamma': 0.1, 'learning_rate': 0.15, 'max_depth': 7, 'n_estimators': 250}
```

```
# Apply the best_params to the model
booster2 = xgb.XGBClassifier(learning_rate=0.15, max_depth=7, n_estimators=250,
                             colsample_bytree=1, gamma=0.1, random_state=17)
booster2.fit(X_train2, y_train2)
run_model(booster2, X_train2, X_test2, y_train2, y_test2, X_validation2, y_validation2)
```

```
Training Accuracy:   98.06%
Test Accuracy:       69.61%
Validation Accuracy: 67.96%
```