COVID-19 Machine Learning

Eric Bae, Ruslan Mukhamedvaleev, Derek Zhu

https://github.com/EricBae21/Analysis-of-COVID-19.git

Medicine Route: Using Machine Learning Modelling for Covid Inpatient Beds grouped by ‘State’ and ‘City’

WHY WE CHOSE THIS TRACK

We noted the ubiquitous impact COVID had on everyone's lives on various levels and wanted to do something about it. We hope that our machine learning model could help reduce the stress of the health care workforce who's facing significant stress and challenges supporting the rest of the community.

TABLE OF CONTENTS

Abstract

As we explored the dataset we were continuously frustrated with the low precision and recall of our models but using the *bootstrap aggregation (bagging)* algorithm, we were able to triple our accuracy.

01

The Results

Our Results and Description of Model

After the exploration of data in our Preliminary Research (See Section 3), our machine learning model predicted correctly 86% of the time!

We first tried using Linear Regression and Logarithmic Regression for our predictions but our accuracy was much too low for any results (around 33%). With that in mind, we found a much better model, the Bootstrap Aggregation (Bagging) Algorithm. This improved both the accuracy and stability of our model by a factor of three. We believe this algorithm had such dramatic effects on our model because training several sample subsets and combining these predictions for the final prediction helped reduce variance in our previous model.

See the next slides (Section 2) for the process, code, data, and quality of our model

02

How We Found It

Technologies Used

All of our work was done with Jupyter Notebook and Python within VS Code.

We used the following libraries:

1. Pandas
2. Sklearn
3. Numpy
4. Matplotlib
5. Seaborn
6. Plotnine
7. Os

Step 1 - Preparing the Data

Files: ‘bootstrap_state.ipynb’ and ‘bootstrap_city.ipynb’

To prepare the data we did the following:

1) First grouped the 742k lines of data by ‘state’ and ‘city’ respectively

```
data = pd.read_csv(
  os.path.expanduser(
    '../Data/COVID-19_Reported_Patient_Impact_and_Hospital_Capacity_by_Facility.csv'
  )).head(800000).sort_values(['state'])
data.head()
```

2) Got rid of all columns that were not floats because we wanted only numbers

```
clean = data
for col in clean.columns:
  if clean[col].dtype != 'float64':
    clean = clean.drop(col, axis=1)
clean = clean.drop(['zip', 'fips_code'], axis=1)
clean
```

Step 1 - Preparing the Data

Files: 'bootstrap_state.ipynb' and 'bootstrap_city.ipynb'

3) Removed all NaN rows

4) Deleted all occurrences of rows with the values '-999,999'

```
def ifZero(x):
    if x < 0:
        return 0
    return x
```

```
clean = data
for col in clean.columns:
  if clean[col].dtype != 'float64':
    clean = clean.drop(col, axis=1)
clean = clean.drop(['zip', 'fips_code'], axis=1)
clean
```

5) Reshaped columns so that it was 2D for the Model

```
output = clean['inpatient_beds_used_covid_7_day_sum'].values.reshape(-1, 1)
clean = clean.drop('inpatient_beds_used_covid_7_day_sum', axis=1)
'inpatient_beds_used_covid_7_day_sum' in clean
```

```
input = clean[col].values.reshape(-1, 1)
```

Step 2 - Testing Features

Files: ‘bootstrap_state.ipynb’ and ‘bootstrap_city.ipynb’

We tested every feature inside of the dataset. We did this by training 70% of the data inside each feature and using 30% for testing. Then we found the accuracy of each feature. Because we wanted to know ‘inpatient_beds_used_covid_7_day_sum’, this was our output for every test and a column in ‘clean’ was the feature we were testing

```
solve = []
maxTrainAccuracy = -1
maxTestAccuracy = -1
index = -1
for col in clean:
  input = clean[col].values.reshape(-1, 1)
  xtrain, xtest, ytrain, ytest = train_test_split(input, output, test_size=0.3, random_state=404)
  dtree = DecisionTreeClassifier(random_state=404)
  dtree.fit(xtrain, ytrain)
  y_pred = dtree.predict(xtest)
  trainAccuracy = accuracy_score(y_true = ytrain, y_pred = dtree.predict(xtrain))
  testAccuracy = accuracy_score(y_true = ytest, y_pred = y_pred)

  if testAccuracy > maxTestAccuracy:
    maxTrainAccuracy = trainAccuracy
    maxTestAccuracy = testAccuracy
    index = len(solve)

  solve.append({
    'col': col,
    'input': input,
    'output': output,
    'xtrain': xtrain,
    'xtest': xtest,
    'ytrain': ytrain,
    'ytest': ytest,
    'dtree': dtree,
    'y_pred': y_pred,
    'trainAccuracy': trainAccuracy,
    'testAccuracy': testAccuracy,
  })
```

Step 3 - Finding the Most Optimal Feature

Files: ‘bootstrap_state.ipynb’ and ‘bootstrap_city.ipynb’

We out of all the features we tested, we found that the *‘total_adult_patients_hospitalized_confirmed_and_suspected_covid_7_day_sum’* was the best predictor. Our prediction rate turned out to be 86%! Our prediction rate for both hospitals organized by ‘state’ and city’ was roughly the same.

```
from math import sqrt

print('name:',solve[index].get('col'))
print('index:',index)
print('max training accuracy',maxTrainAccuracy)
print('max test accuracy',maxTestAccuracy)

interval = 1.96 * sqrt((maxTestAccuracy * (1 - maxTestAccuracy)) / len(solve[index].get('ytest')))
print('margin of error:',interval)

print(maxTestAccuracy,'+-',interval)
```

[11] ✓ 0.0s

```
name: total_adult_patients_hospitalized_confirmed_and_suspected_covid_7_day_sum
index: 25
max training accuracy 0.8714067914067914
max test accuracy 0.8612727978269306
margin of error: 0.005448468144899889
0.8612727978269306 +- 0.005448468144899889
```

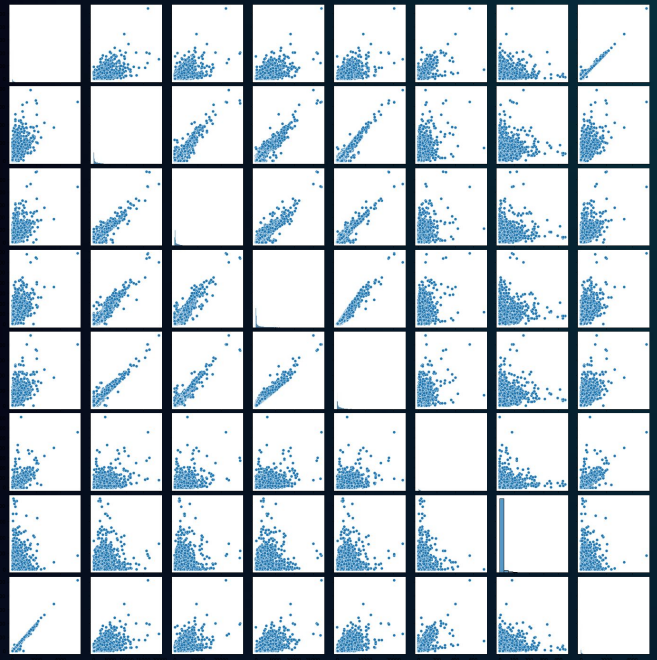
03

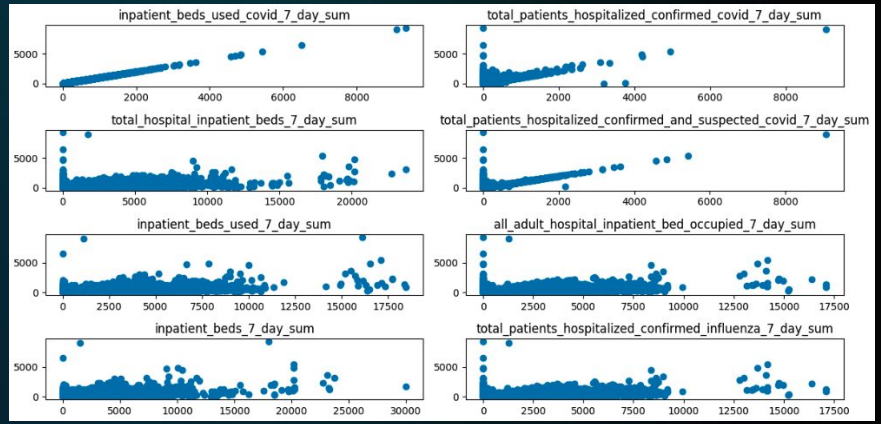
Preliminary Results

Exploration of Data

Files: ‘dAnalysis.ipynb’ and ‘eric.ipynb’

Before we started on creating our model and predictions, our team spent a good amount of time exploring the dataset and looking at all the patterns to better understand it.





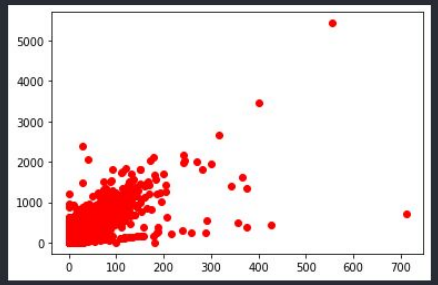
Determining Correlation

Files: ‘dAnalysis.ipynb’

We spent a lot of time searching for trends that showed high correlation between values. We started with thinking in linear terms, where we searched for correlation coefficients that could help determine our model’s features.

```
plt.scatter(X_5, y, color='red')
```

```
<matplotlib.collections.PathCollection at 0x21f90f90490>
```




```
np.corrcoef(X_5, y)
```

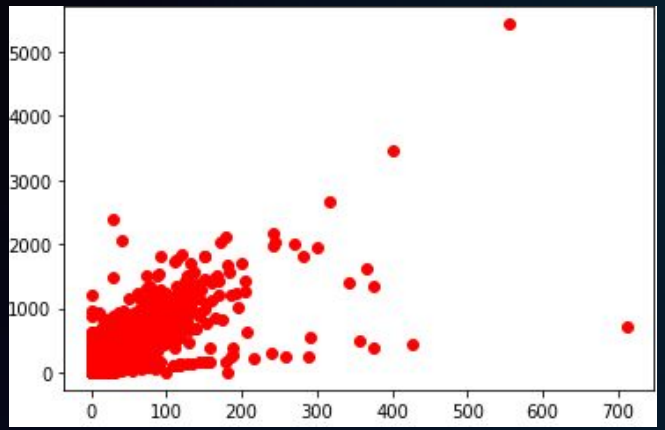
```
array([[1.        , 0.79406297],
       [0.79406297, 1.        ]])
```

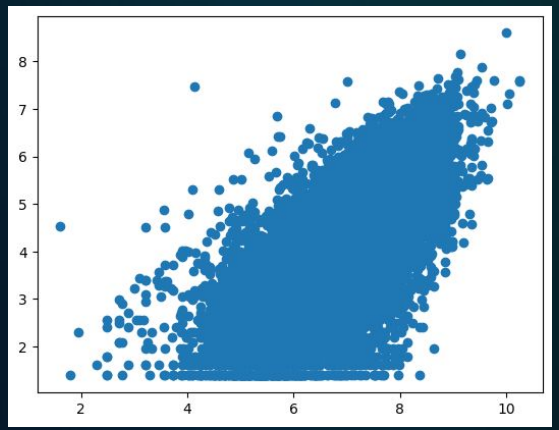
By using scripts to find correlation coefficients for a multiple regression model, we realized that many of the columns would introduce a lot of noise into our predictive model.

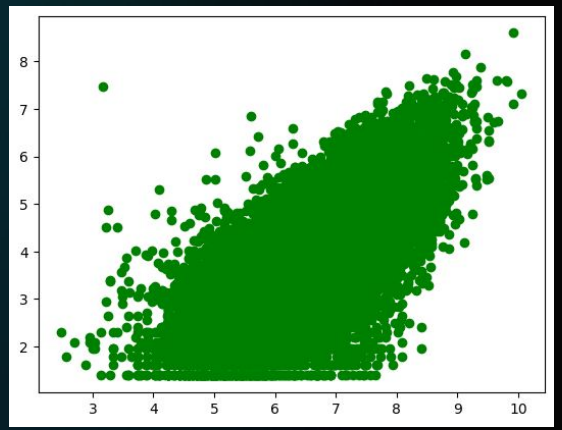
Correlation in our first model

Files: ‘dAnalysis.ipynb’

Our first model attempt used multiple linear regression trends to attempt to find a suitable output. We took three of the highest correlated columns: `previous_day_admission_adult_covid_confirmed_7_day_sum`, `total_beds_7_day_sum`, and `all_adult_hospital_inpatient_beds_7_day_sum`. We normalized the data and used a linear regression model for prediction. Below are the graphs of the three variables plotted against inpatient_beds_used_covid_7_day_sum respectively





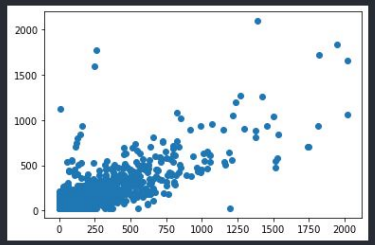


Results of first model

The result of our first multiple linear regression model was poor. This lead us to go back to the drawing board and evaluate other options in feature extraction or model selection. Below are the results of our predictive model:

```
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x21f8dee23b0>
```




```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
metrics.r2_score(y_test, predictions)
```

```
MAE: 48.431934776609914
MSE: 10396.59716072846
RMSE: 101.96370511475375

0.670743834905041
```

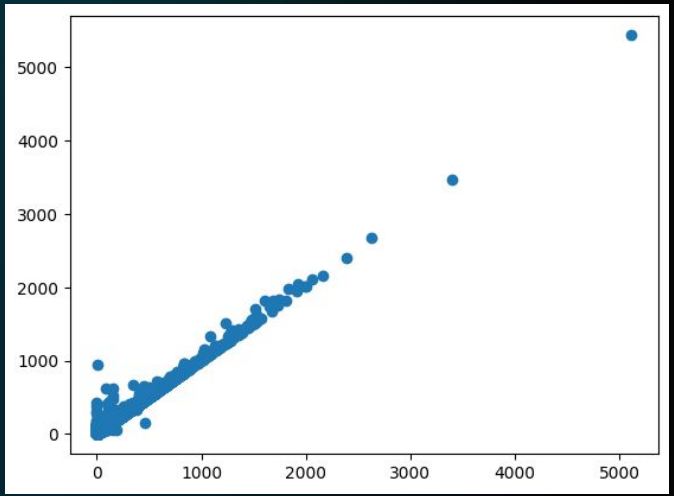
Feature extraction

Files: ‘dAnalysis.ipynb’

In this photo where X is: `total_adult_patients_hospitalized_confirmed_and_suspected_covid_7_day_sum`

And y is: `inpatient_beds_used_covid_7_day_sum`

We determined that the X feature had the most linear correlation in determining the amount of beds that would be used. Thus we would end up using this in our regression model.



04

Our Conclusion

Conclusion

Before we started working with the data set, we filtered and cleaned the data so it suited our needs the best. To explore the data, our team spent a significant amount of our time creating subsets and plotting scatter plots and subplots to understand how the data was organized. After we understood the data better, we scrapped our initial plans of using Linear and Logarithmic Regressions and decided to use the Bootstrap Aggregation Algorithm to increase the stability and accuracy of our model with a margin of error of +/- 0.05. We believe that this model is a good predictor of inpatient beds based on Covid. Because of this, we believe we hit every single criteria of the Machine Learning Judging Rubric.

THANKS

We had fun at the Datathon.

Please let us know if you have any questions or comments about our code, work, or slides. We would be happy to answer them

- Eric, Ruslan, Derek