Covid-19 Awareness and Covid-19 Cases in Ohio

Aakash Chaudhari [1] and *Pradhyumna Rao* [2]

Abstract

To understand the Essence of the Impact of Covid 19 in Ohio and the Various parameters that have affected the state right from the number of hospitalizations, cases and deaths. We aim to train a model that predicts the number of cases based on all of the remaining variables in the dataset, and report the R2-Value of your best model.

Data

We divided the Data Analysis in Four Parts :

1. Data Collection - The awareness data has been extracted from tweets and has been collected by using hashtags posted by users in Ohio during the Covid-19 pandemic. To prepare the dataset, over 46,000,000 million tweets posted by over 91,000 have been collected. Here are the steps used to create the database.

2. Data Preparation - While preparing the Data we analyzed the important features that would help us to build the Model, using the similarity index.

3. Data Analysis - From the county wise data we analyzed that Maximum cases are from Franklin and Minimum being from Monroe and Pilce.

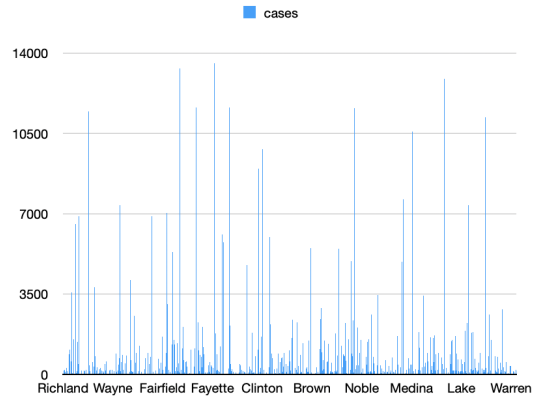

Figure 1 : County v/s Cases

4. Data Interpretation - Using this Data we move ahead with extracting the important features and choosing the best model.

Keywords **: Model, Data Analysis, Accuracy , ML**

I. INTRODUCTION

For the given data, we have around 140 features some of which are normalized, some features include health, sports, politics, cases, illness, age, poverty etc.
Treating it as a regression problem, we study the data and derive insights from it in ogre to gain data and knowledge. Our approach is to study the Covid impact in Ohio and how much damage the pandemic had with respect to cases and the methods to reduce the impact. Using the data we are given along with various features and using machine learning models we predict the covid impact.

Using the data we have and first preprocess. We look at the shape, features and try to understand which features are important. Keeping the main focus on model accuracy, we realize PCA would help us to improve model performance. Doing feature extractions and applying ML models help us to improve accuracy.

For the scope of our project, We will be using the Data available and using the Data we will try various models so as to find the Perfect r2 score. Finding the best model fit, we will try to optimize the model and make sure our model works on the Test data available.

II. METHODS

To gather insights, we will preprocess the Data and use PCA technique. We used PCA to reduce the dimensionality of the data, We are trying to capture as much variance as possible. We tried to achieve 95% of variance and based on that we got 17 Features. This completes the main preprocess we aim to achieve.

We see the data has cases of each county day wise. For each county if we look that the number of days is less than 80, the number of cases is zero. The cases start to rise after Day 80.

Having the data with features that we think are useful we perform normalization.After seeing the Number of Threshold that covers 95% of variance we choose 5 PCA components that we believe would cover the data and explain most of the information.

The next method is to use binning to convert continuous variables to Categorical variables using Decision tree Regressor. By breaking down the data into smaller and smaller subsets give us an idea of the distribution.

With the aim to increase the model fit we tried using the Approach of binning, transforming numeric features into categorical features, using a set of thresholds. We realize that Binning of continuous variables introduces non-linearity and tends to improve the performance of the model.

For proper model selection we use pycaret that helps in understanding Machine Learning Flow. Compared with the other open-source machine learning libraries, PyCaret is an alternate low-code library that can replace hundreds of lines of code with a few lines only. This makes experiments exponentially faster and more efficient. PyCaret is essentially a Python wrapper around several machine learning libraries and frameworks such as scikit-learn, XGBoost, LightGBM, CatBoost, spaCy, Optuna, Hyperopt, Ray, and a few more.

To further enhance the features, we used feature selection Algorithm, Mainly RFE : Recursive Feature Elimination. Features (columns) that are the most pertinent to a dataset are chosen using techniques known as feature selection. Machine learning algorithms can operate more effectively (less space or time complexity) and efficiently (fewer features).

As we wanted to choose those Features in the training dataset that are most relevant in predicting the Target variable. The thing which helps the most is the number of features to be included as well as the model to be selected. Both of this is

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	57.1167	151845.5161	333.1299	0.7927	0.9096	0.6761	0.7650
gbr	Gradient Boosting Regressor	76.1528	136191.3060	314.5276	0.7894	2.1553	1.2350	0.3920
xgboost	Extreme Gradient Boosting	59.4997	160846.0399	341.3881	0.7883	1.1709	0.6838	0.3790
rf	Random Forest Regressor	60.0295	153833.1242	331.5959	0.7784	0.9428	0.7494	0.8310
lightgbm	Light Gradient Boosting Machine	84.7413	164219.7890	364.1678	0.7227	1.9807	1.4725	0.7110
dt	Decision Tree Regressor	70.5920	242687.6437	424.4586	0.5619	1.0327	0.8535	0.0850
ada	AdaBoost Regressor	177.0276	247985.0944	455.2861	0.5015	3.5781	4.7491	0.4330
knn	K Neighbors Regressor	104.1376	318523.0780	522.0750	0.4319	1.3711	1.4912	0.0860
huber	Huber Regressor	123.2954	557770.4678	660.8406	0.3034	1.5520	1.5507	0.2490
br	Bayesian Ridge	252.2358	465236.7656	631.3861	0.2320	3.9956	5.7865	0.0440
lasso	Lasso Regression	253.2916	464973.0955	632.1490	0.2242	3.9842	5.8427	0.0470
llar	Lasso Least Angle Regression	253.2902	464972.6275	632.1486	0.2242	3.9842	5.8427	0.0460
en	Elastic Net	257.5348	529381.1858	662.5794	0.2223	4.1410	6.7631	0.0430
ridge	Ridge Regression	255.3611	465256.6683	632.6891	0.2207	4.0031	5.9123	0.0410
lr	Linear Regression	255.6154	465273.7110	632.8036	0.2198	4.0032	5.9225	0.3650
omp	Orthogonal Matching Pursuit	271.4252	631979.4057	727.2939	0.0518	3.9931	10.2219	0.0420
dummy	Dummy Regressor	259.3121	697099.6081	761.8082	-0.0183	4.3450	3.8829	0.1350
lar	Least Angle Regression	385.3840	540414.0594	704.9879	-0.2052	4.5834	9.5115	0.0450
par	Passive Aggressive Regressor	290.9415	609499.8067	725.1656	-0.3725	3.8623	6.7883	0.0570

Fig 1 : Best Models Comparison

From all the EDA and preprocessing we came to the conclusion to apply the ExtraTree Regressor Model in our preprocessed Data. Extra trees (short for extremely randomized trees) is an ensemble supervised machine learning method that uses decision trees and is used by the Train Using AutoML tool.

The extra trees algorithm, like the random forests algorithm, creates many decision trees, but the sampling for each tree is random, without replacement. This creates a dataset for each tree with unique samples. A specific number of features, from the total set of features, are also selected randomly for each tree. The most important and unique characteristic of extra trees is the random selection of a splitting value for a feature. Instead of calculating a locally optimal value using Gini or entropy to split the data, the algorithm randomly selects a split value. This makes the trees diversified and uncorrelated. [5]

Now with the model that we have, we further look to improve the performance of the Model using hyperparameters. The parameters we use :-

```
n_estimators=1000, max_depth=7,
eta=0.1, subsample=0.7,
colsample_bytree=0.8
```

XGBoost is a special implementation of a gradient boosting machine that uses more accurate approximations to find the

best model. It improves upon gradient boosting machine framework through systems optimization and algorithmic enhancements. [6]

These are the Estimators, max_depth, learning rate etc. The XGBRegressor from the XGBoost library is used to build a regression Model from gradient Descent. In our case the hyperparameters we used helped in improving the Accuracy level and improving r2 score.

```
model.fit(X_train, y_train)

Fitting 2 folds for each of 9
[14:10:20] WARNING: ../src/obj
[14:10:20] WARNING: ../src/lea
Parameters: { "silent" } are n
```

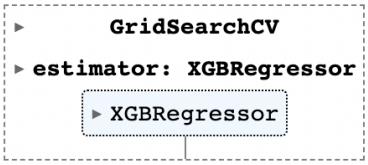

Fig 2 : XGBRegressor

To further enhance the Model, we used Neural Networks. Using an experimental approach for multiple columns, we used Epochs - 400 and Learning Rate - 0.001, keeping the Batch size as 8.
Using a linear layer that performs a linear transformation on the input data. Using the Relu Activation function. We use 4 linear layers with the first one is an input layer, two other layers are hidden and the last one is an output layer. We experimented using features selected using Rfe from the Whole data. For Adaptive Learning Rate , we used Adam's optimization Algorithm which is a further extension of stochastic gradient descent to update network weights during training.

V. Results

To identify the best models we used pycaret which gave the best model to choose from. [Fig 3].

Following which we apply the Model we selected which is ExtraTreeRegressor. We get the results in [Fig 4].

To enhance the model more we apply XGBRegressor [Fig 5].

Following which we get an r2 score of **0.89883** in test data.

	fit_time	score_time	test_score	train_score
0	0.351971	0.017689	0.781131	0.995874
1	0.348204	0.016921	0.914963	0.990377
2	0.345601	0.016982	0.952988	0.991996
3	0.355223	0.017030	0.965051	0.994185
4	0.353234	0.024352	0.936444	0.995294

Fig 3 - Pycaret Model

	fit_time	score_time	test_score	train_score
0	4.071477	0.023184	0.744993	0.995772
1	2.628087	0.025593	0.931601	0.990262
2	2.573905	0.023158	0.948504	0.991879
3	6.931241	0.022931	0.938328	0.994086
4	2.505363	0.023014	0.939807	0.995238

Fig 4 - ExtraTreeRegressor

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
xgboost	Extreme Gradient Boosting	53.4047	129697.2926	293.1806	0.8766	1.0050	0.6523
rf	Random Forest Regressor	54.1999	128274.8771	297.9548	0.8152	0.9089	0.6163
dt	Decision Tree Regressor	53.2364	132860.3881	311.2617	0.8054	0.9665	0.6624
gbr	Gradient Boosting Regressor	70.6973	126008.5239	302.9653	0.7998	2.0595	1.1662
et	Extra Trees Regressor	51.6571	132151.2125	309.4075	0.7861	0.8284	0.5851
lightgbm	Light Gradient Boosting Machine	78.6988	161846.8286	360.2755	0.7228	1.6992	1.3101
ada	AdaBoost Regressor	117.3235	169413.4436	366.7105	0.7034	3.0067	2.3345
knn	K Neighbors Regressor	87.0960	251644.3965	454.8824	0.3298	1.1426	0.8816
huber	Huber Regressor	121.0335	553414.9996	658.4710	0.3079	1.1658	1.5727
par	Passive Aggressive Regressor	182.3271	541973.3620	659.1357	0.2817	3.4382	3.7568
br	Bayesian Ridge	242.0342	471111.9319	634.5460	0.2298	3.8820	5.5724
lasso	Lasso Regression	242.1022	471156.3548	634.9076	0.2268	3.8734	5.6005
llar	Lasso Least Angle Regression	242.1020	471155.7189	634.9071	0.2268	3.8734	5.6005
ridge	Ridge Regression	243.4005	471153.1022	635.1986	0.2244	3.8808	5.6520
en	Elastic Net	252.9464	537999.1203	666.0206	0.2244	4.1290	6.1378
lr	Linear Regression	243.6430	471169.2076	635.3188	0.2234	3.8799	5.6660
omp	Orthogonal Matching Pursuit	237.5698	644459.0144	732.3246	0.0543	3.4703	9.5703
lar	Least Angle Regression	342.2473	506204.8168	673.5049	0.0233	4.3747	8.9173
dummy	Dummy Regressor	259.3121	697099.6081	761.8082	-0.0183	4.3450	3.8829

Fig 5 - XGBRegressor

Bibliography

1. https://www.datacamp.com/tutorial/guide-for-automating-ml-workflows-using-pycaret
2. https://machinelearningmastery.com/rfe-feature-selection-in-python/
3. https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-extra-tree-classification-and-regression-works.htm
4. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
5. https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-extra-tree-classification-and-regression-works.html
6. https://www.geeksforgeeks.org/xgboost/