README Code File Doc: In this folder are the code files for the COVID-19 Subvariant Spike Prediction group. We sought to predict the number of weeks for a subvariant to reach its peak spike by state and county. Below is a description for all of the code files provided in this folder grouped by category.

Merging, Cleaning, Calculating Data:

Creating State Data.ipynb - This code creates part of the state data.

1. Read in required libraries for analysis
2. Read in Johns Hopkins Case data
3. Calculate county daily cases from cumulate case number
4. Sums cases by week by state
5. Merges weekly case data with CDC variant proportion data, multiplies them together to get the number of weekly cases by subvariant
6. Finds the weeks where the first case happened by state by variant and finds the weeks where the maximum # of cases happened by state by variant and subtracts them to get the difference (target variable)
7. Export to CSV

Creating County Data.ipynb - This code creates part of the county data.

1. Read in required libraries for analysis
2. Read in Johns Hopkins daily Case data
3. Sums cases by week
4. Merges weekly case data with CDC variant proportion data, multiplies them together to get the number of weekly cases by subvariant
5. Finds the weeks where the first case happened by county by variant and finds the weeks where the maximum # of cases happened by state by variant and subtracts them to get the difference (target variable)
6. Merge county demographic and transportation features
7. Create dummy variables
8. Export to CSV

Adding Features - State Data.ipynb - This code adds features to state data

1. Read in required libraries for analysis
2. Read in covid data files and feature data files
3. Create dummy variables
4. Drop variants we are not predicting
5. Merge transportation, airport, road type data by state
6. Export to CSV

Merge_demo_data.py - This code adds features to the state data

1. Import libraries
2. Read in data files

3. Merge all data files to main file
4. Export to csv

Jupyter merge airport.py - This code adds features to the state data

1. Read in libraries
2. Read in data files
3. Clean data
4. Merge new data to main file
5. Export to csv

Evaluating Models - Hyperparameter Tuning:

Grid Search for RF, DT, and SVR State Data.ipynb. - Performing GridSearchCV to find best hyperparameters for Random Forest, Extra Regressor Tree, and SVR for state data

1. Read in libraries
2. Read in state data
3. Create X and Y variables
4. Split data using group test train
5. Perform GridSearch CV using GroupKFold on all models to get the best hyperparameters
6. For SVR, create pipeline to scale the data when performing GridSearch
7. Evaluate the effect of Random Forest feature selection on SVM

Parameter Tuning SVR-County.py - Tuning the hyperparameters of SVR using county data

1. Read in libraries
2. Read in county data
3. Create X and Y variables
4. Split data using group test train
5. Create pipeline to scale the data when performing GridSearch Cross Validation and get best hyperparameters
6. Evaluate the effect of Random Forest feature selection on SVM

Tuning RF hyperparameters - county data.py - tuning the hyperparameters for random forest using county data with GridSearchCV and performing cross validation

1. Read in libraries
2. Read in county data
3. Create X and Y variables
4. Split data using group test train
5. Perform GridSearch CV using GroupKFold on RF to get the best hyperparameters
6. Calculate Feature Importance

Tyler Models - Grid Search DTs - county data.py - tuning hyperparameter of extra tree regressor using county data with GridSearchCV

1. Read in libraries
2. Read in county data

3. Create X and Y variables
4. Split data using group test train
5. Perform GridSearch CV using GroupKFold to get the best hyperparameters

County Level_COVIDSpike_XGBoost.py and **State_Level_COVID_XGBoost.ipynb**(ReadMe for both files) - these files find the best hyperparameters for XGBoost using state and county data

1. Read in required libraries for analysis
2. Read in state/county data
3. Create X and Y variables
4. Split training and testing sets
5. Create grid search CV for hyper parameter tuning
6. Get the best parameters to use in the final model evaluation

CountyData_Tree_ExtraTree_SVR.ipynb - Additional work done on Extra Trees and SVM for hyperparameter tuning using county data

1. Read in required libraries for analysis
2. Read in county data
3. Create X and Y variables
4. Split training and testing sets
5. Run Decision Tree Regressor, get metrics and plot
6. Run Extra Tree Regressor, get metrics and plot
7. Scale data with MinMaxScaler for SVR
8. Run SVR, get metrics and plot
9. Create grid search CV for hyper parameter tuning
10. Get the best parameters to use in the final model evaluation

StateData_Tree_ExtraTree_SVR.ipynb - Additional work done on Extra Trees and SVM for hyperparameter tuning using state data

1. Read in required libraries for analysis
2. Read in state data
3. Create X and Y variables
4. Split training and testing sets
5. Run Decision Tree Regressor, get metrics and plot
6. Run Extra Tree Regressor, get metrics and plot
7. Scale data with MinMaxScaler for SVR
8. Run SVR, get metrics and plot
9. Create grid search CV for hyper parameter tuning
10. Get the best parameters to use in the final model evaluation

Evaluating Results Using Best Hyperparameters For Each Model:

Running best parameters of every model - county data.py - using the best hyperparameters for every model using county data and running the models on the test data to get final results.

1. Read in libraries
2. Read in county data
3. Create correlation matrix and drop highly correlated features
4. Split data using group test train
5. Get results using test data for each model before tuning hyperparameters
6. Get results using test data for each model after tuning hyperparameters
7. 10-fold CV using training data to understand generalizability for each model
8. Calculate feature importance for RF and XGBoost
9. Get results using training data for each model after tuning hyperparameters

Running best parameters of every model - state.py - using the best hyperparameters for every model using states data and running the models on the test data to get final results.

1. Read in libraries
2. Read in state data
3. Create correlation matrix and drop highly correlated features
4. Split data using group test train
5. Get results using test data for each model before tuning hyperparameters
6. Get results using test data for each model after tuning hyperparameters
7. 10-fold CV using training data to understand generalizability for each model
8. Calculate feature importance for RF and XGBoost

Additional Work Not Included in Results:

Predicting number of max cases at peak spike.py - Additional work not included in results; modeling to predict the max number of cases at peak spike.

1. Read in libraries
2. Read in county data
3. Split data using group test train
4. Get results using test data for each model before tuning hyperparameters
5. Calculate feature importance of RF and XGBoost
6. Read in state data
7. Split data using group test train
8. Get results using test data for each model before tuning hyperparameters
9. Calculate feature importance of RF and XGBoost

Statedata_NeuralNetwork.ipynb - not included in results, but an attempt to use neural networks for state level prediction

1. Read in required libraries for analysis
2. Read in state data
3. Create X and Y variables
4. Split training and testing sets
5. Scale data with MinMaxScaler for MLPRegressor Neural Network

6. Run MLPRegressor , get metrics and plot
7. Create grid search CV for hyper parameter tuning
8. Get the best parameters to use in the final model evaluation