Exploring the use of Machine Learning to Estimate Transit Ridership Based on Socio-Demographic Variables and Transportation System Characteristics

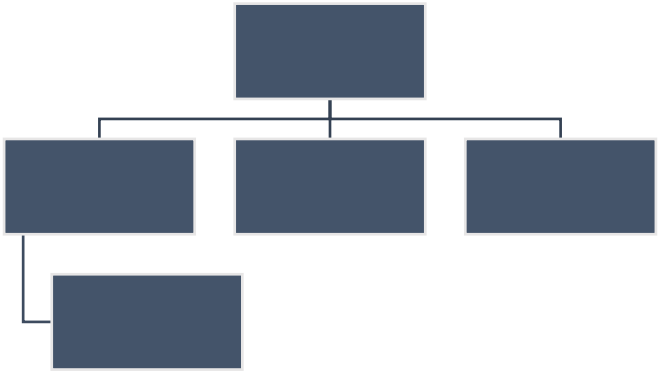
Model Development and Validation

!

Natalia Ruiz Juri: nruizjuri@mail.utexas.edu
Gopindra Nair: gopindra@utexas.edu

This document is part of a package that contains the scripts for developing machine learning models that predict transit ridership along bus-routes in the Dallas-Fort Worth region. The folder structure of the package and the tools required for running the scripts are described. Following this, the preprocessing steps for generating the final dataset used for training and testing the machine learning models are briefly explained. Finally, the machine learning algorithms that were used and the model results are discussed.

The project folder structure is as shown in Figure 1. The parent folder, contains three folders – , and . The folder structure with the initial set of files (before any of the script files are executed) is shown in more detail in Section A1 of the Appendix.



Within the folder, another folder named stores all the input data files required for this project. More information on the input data files is provided in Section 2. The folder will be populated with more data files that are derived from the input data files as the data processing scripts are executed.

The folder contains all the scripts that are used for preprocessing, model estimation and prediction. The scripts are written in the Jupyter Notebook format and have the

extension IPYNB. Files of this format may be opened and executed in software such as and . The folder also contains a folder named which has the file. is a python file that contains the code for a GIS function that translates geographical attributes (or polygon features) from one shapefile geometry to another. This function in the is called by some of the IPYNB script files in the folder. The folder is initially empty. When the script in the folder is executed, the folder will be populated with plots of the goodness-of-fit and a file that stores the predicted ridership information.

All the scripts used for this project were developed using the Anaconda distribution of Python. Specifically, Python 3.7 was used as the interpreter. The steps for installing Anaconda and all the libraries required for running the scripts developed for this project are as follows.

1. Download and install the most recent version of Anaconda from: https://www.anaconda.com/products/individual.
2. The installation procedure would have installed a program named and a program named . First, open the program . provides a command-line interface for accessing Python development tools and for managing the Python development environment. Figure 2 shows the interface.

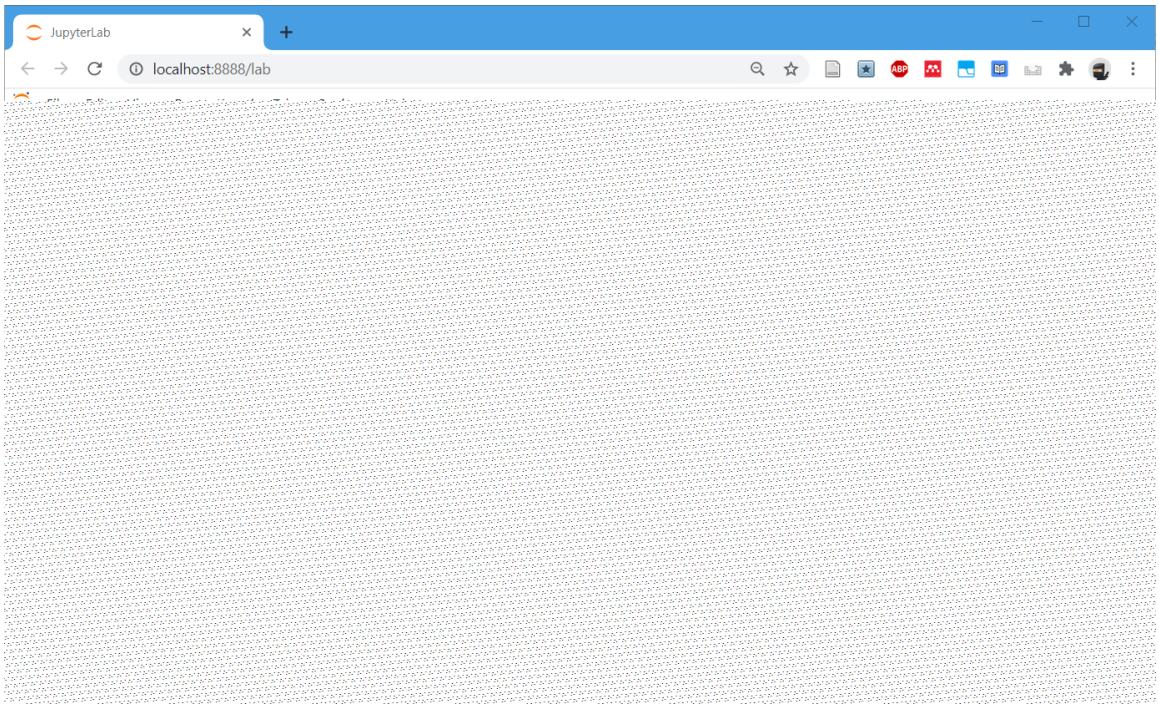
3. This project requires the installation of several Python packages in addition to those that are included in a standard installation of Python. When installing packages specific to a project, it is advisable to do so in a separate Python environment. Therefore, in this step, create a new Python environment named "Transit_Prediction" based on Python 3.7 using the following command:

4. Activate the newly created Python environment using the command:

5. Install the packages required for this project by executing the following commands:

6. Open Jupyter Lab using the command:

   The above command would also start a Jupyter server in the background. Jupyter Lab will be opened in the default internet browser. The interface of Jupyter Lab is as shown in Figure 3. All the scripts used in this project can be accessed by double-clicking on the script name in the tab on the left side.



7. To quit Jupyter lab, select -> and then close the JupyterLab tab in the browser. To open Jupyter Lab the next time, simply open and follow instruction 4 and instruction 6.

The procedure mentioned above for installing Python libraries and accessing Jupyter notebooks uses the command-line tool. Alternatively, the user could also use the tool which has a Graphical User Interface for performing the same operations. Instructions for using the can be found at https://docs.anaconda.com/anaconda/navigator/getting-started/.

the number of houses in the census tracts. All in all, a total of 218 socio-demographic and employment counts are computed at the census tract level.

This notebook computes the distance buffers for each transit route and aggregates the socio-demographic and employment counts within the buffers. A buffer of a transit route is the geometry of the area that is within a threshold distance from any point along the transit route. The aggregated socio-demographic counts for a buffer is computed by taking the sum of socio-demographic counts of the census tracts that overlap with the buffer. If a census tract overlaps only partially with the buffer, the socio-demographic count of the census tract is multiplied by the fraction of the area of overlap of the census tract before adding it to the buffer. The aggregated employment counts for a buffer area are also calculated similarly except that TSZs are considered instead of census tracts.

The aggregated feature counts are computed for buffer distances of 200m, 400m, 800m and 1600m. Additionally, the aggregated features counts are also computed for the buffer bands of 200-400m, 400-800m and 800-1600m by differencing the feature counts in the smaller distance buffer area from the feature counts in the higher distance buffer area. For example, the aggregated feature count within the buffer band of 400-800m will be the difference between the aggregated feature count in the 400m buffer and the aggregated feature count in the 800m buffer. In this manner, for each socio-demographic/employment related census-tract/TSZ level feature, a total of seven aggregated features (4 for full distance buffers and 3 for banded buffers) are computed, which results in a total of 218 x 7 = 1526 aggregated feature counts for each route.

This notebook computes the schedule related characteristics of each Transit route using the pickled GTFS data. The notebook computes the frequency of routes, which is the average number of arrivals of buses of a given route at stops along the route divided by the duration of the time period under consideration (in hours). The frequency is computed for the 24-hour period of a day as well as for the five time periods of,

1. AM Peak – 6:30 AM to 9:00 AM
2. Midday 1 – 9:00 AM to 12:00 noon
3. Midday 2 – 12 noon to 3:00 PM
4. PM Peak – 3:00 PM to 7:00 PM
5. Night – 7:00 PM to 6:30 AM (next day)

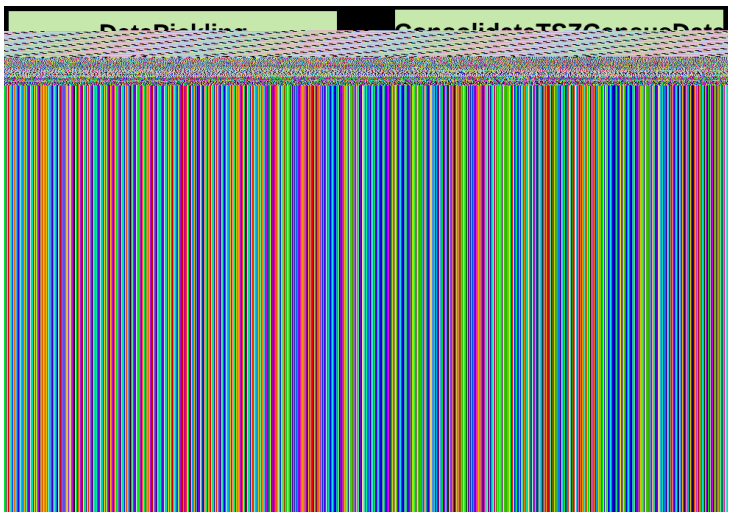
This notebook also computes the number of unique stops along a route.

This notebook produces the final dataset with the features and the dependent variable that can be used for fitting machine learning models. It combines the features computed in and . It computes the dependent variable of daily ridership along a route using the survey dataset.

This notebook fits the different machine learning models and evaluates goodness-of-fit. The tasks performed in this notebook are explained in detail in Section 4.

Some of the notebooks take as input the output files generated by other notebooks. For example, the notebook can be executed only after the dataset used for fitting the models has been produced by the notebook . The flowchart in Figure 5 shows the dependencies between the notebooks. The notebook at the head end of an arrow should be executed only after the notebook at the tail end has been executed. The notebooks highlighted in green are not dependent on any other notebooks. Therefore, the sequence of execution can begin from these notebooks. As the notebook files are executed, the folder will be populated with the final dataset for estimation and intermediate datasets that were used for processing. All the results will be saved in the folder.

The specific input and output files for each of the notebooks are provided in Section A2 in the Appendix. The folder structure after all the scripts in the folder have been executed will be as shown in Section A3 in the Appendix.



More than 1500 features were computed for each transit route. Since the use of a large number of features can result in less accurate machine learning models (with a tendency to overfit), only a subset of 41 features was selected for use in the machine learning models. The selected features are shown in Table 1. Features that relate to combinations of multiple socio-demographic characteristics of households (which formed a majority of the available features) were not

b02_HH_W2	No. of HHs with 2 workers within 200 meters from the route
b02_HH_W3	No. of HHs with 3+ workers within 200 meters from the route
b0204_HH_W1	No. of HHs with 1 worker between 200 and 400 meters from the route
b0204_HH_W2	No. of HHs with 2 workers between 200 and 400 meters from the route
b0204_HH_W3	No. of HHs with 3+ workers between 200 and 400 meters from the route
b0408_HH_W1	No. of HHs with 1 worker between 400 and 800 meters from the route
b0408_HH_W2	No. of HHs with 2 workers between 400 and 800 meters from the route
b0408_HH_W3	No. of HHs with 3+ workers between 400 and 800 meters from the route
b02_HH_C1	No. of HHs with 1 child within 200 meters from the route
b02_HH_C2	No. of HHs with 2+ children within 200 meters from the route
b0204_HH_C1	No. of HHs with 1 child between 200 and 400 meters from the route
b0204_HH_C2	No. of HHs with 2+ children between 200 and 400 meters from the route
b0408_HH_C1	No. of HHs with 1 child between 400 and 800 meters from the route
b0408_HH_C2	No. of HHs with 2+ children between 400 and 800 meters from the route

* ridership is the dependent variable

The complete dataset (after the removal of a route named "The Spur" because it had an unusually high ridership for its length) had 160 routes. A random stratified splitting approach was used to split this dataset into the training and testing datasets. Under this approach, the complete dataset was first divided into quartiles based on ridership. From each quartile, 30% of the routes were randomly selected without replacement and added to the testing dataset and the remaining 70% were added to the training dataset. The resulting testing dataset had a total of 48 routes with 12 routes from each quartile and the resulting training dataset had a total of 112 routes with 28 routes from each quartile. All the machine learning models were trained using only data from the training dataset. The testing dataset was used only to test the accuracy of the models.

Numerical measures and graphical plots were used to assess the Goodness-of-fit of the different machine learning models.

The Coefficient of Determination ( ) was used as a measure of the goodness-of-fit of the fitted models. The is the proportion of variance in the outcome variable that is explained by the predictors. Consider records of the outcome variable indexed as , , …, . Let the predicted outcome for these records be , , …, . Then is computed as follows,

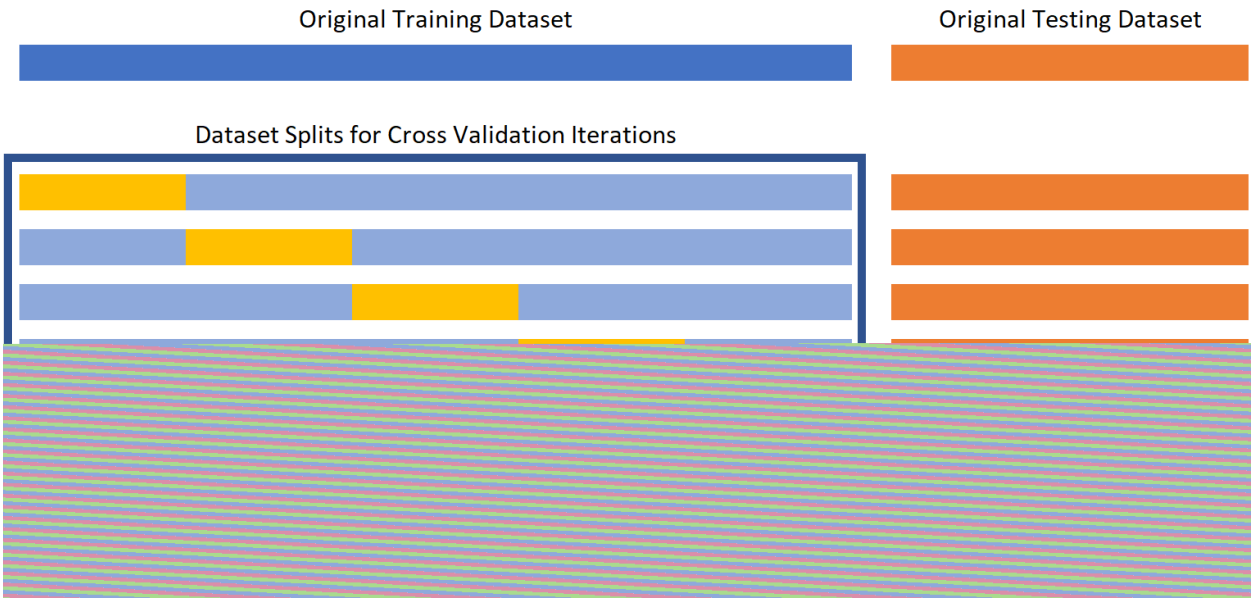
—— , (1)

, (2)

– . (3)

where ¯ is the observed mean of the outcome, is the sum of squared residuals and is the sum of squared totals. For assessing model fit, the following metrics based on were computed.

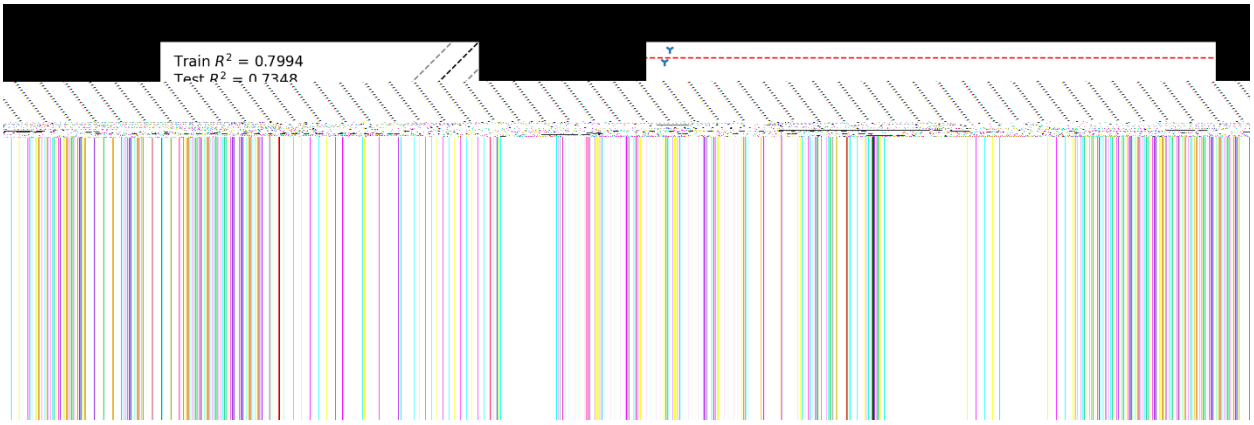
1. Train - where the model is fitted using the training dataset and is computed with respect to the training dataset.
2. Test - where the model is fitted using the training dataset and is computed with respect to the testing dataset. A value for Test that is much lower than the Train could indicate that the machine learning model has been overfitted.
3. Cross Validated - Here, the original training dataset is split into five parts. Then, considering each of the individual parts as a new testing dataset and the remaining four parts as a new training dataset, the model is fitted using only the new training dataset and the value is calculated with respect to the new testing dataset. The cross-validated is the mean of the s obtained for the five new testing datasets. An illustration of the splitting of the original training dataset into the training and testing datasets for cross-validation if provided in Figure 6.



The Train , Test and Cross-Validated values computed by the notebook are summarized in the file in the folder.

technique that is commonly used for hyperparameter tuning. The use of 5-fold cross-validation was already explained in Section 4.3.1 in the context of computing Cross Validated . For hyperparameter tuning, the Cross Validated is computed for a range of hyperparameter values. Finally, the hyperparameters are assigned the values that produce the best Cross Validated . Once the hyperparameters are tuned, the parameters are estimated once again with the complete training dataset (instead of part of the training dataset as done in the cross-validation procedure).

For tuning the parameter, a range of 100 values between ~0.5 to ~550 were tested. Note that the search space was not uniformly split to produce the 100 values. Values in the lower end of the search space were closer to each other than values in the higher end. The best fit was produced by . The goodness-of-fit evaluation plots and metrics for the Lasso Regression model are provided in Figure 9.



The Polynomial Lasso Regression is an extension of Lasso Regression where the feature set used for prediction includes the original features as well as polynomial transformations of the original features. The transformed features can include powers of the original features and products of powers of the original features. For this project, we estimate one model where the features set is expanded by also including the squares of all the original features and another model where the feature set is expanded by including the square roots of all the original features. We did not consider products of the original features in the new feature set because this would cause the new feature set to become extremely large relative to the number of records in the database and therefore more likely to cause overfitting. The goodness-of-fit evaluation plots and metrics for the Lasso Regression model with squares of the features are provided in Figure 10 and that for the Lasso Regression model with square roots of the features are provided in Figure 11. The optimal values for these models were found to be 44.4 and 31.3 respectively.

A2 Inputs and Outputs of Notebook Files

Notebook	**Inputs**	**Outputs**
DataPickling	Data/Inputs/GTFS/DART GTFS (GTFS Folder)	Data/GTFS/routes.pickle
	Data/Inputs/GTFS/FWTA GTFS (GTFS Folder)	Data/GTFS/shapes.pickle
	Data/Inputs/NCTCOG_CONSOLIDATED_DART_THET_DCTA_Sep0315.xlsx	Data/GTFS/stop_times.pickle
		Data/GTFS/stops.pickle
		Data/GTFS/trips.pickle
		Data/GTFS/gtfs_schedule.pickle
		Data/survey_2014.pickle
ConsolidateTSZCensusData	Data/Inputs/Market Segmentation/TAZ_Emp_IGxInd.csv	Data/census_SD.pickle
	Data/Inputs/Market Segmentation/TAZ_HH_SxWxIG.csv	Data/Census_SD (Shape folder)
	Data/Inputs/Market Segmentation/TAZ_HH_WxIGxC.csv	
	Data/Inputs/Market Segmentation/TAZ_HH_WxVxIG.csv	
	Data/Inputs/TSZ5352 (Shape folder)	
	Data/Inputs/dfw_census (Shape folder)	
RouteBufferFeatures	Data/census_SD.pickle	Data/route_buffer_dat.pickle
	Data/GTFS/shapes.pickle	
	Data/GTFS/gtfs_schedule.pickle	
RouteScheduleFeatures	Data/GTFS/gtfs_schedule.pickle	Data/route_schedule_dat.pickle
RouteFeatures	Data/survey_2014.pickle	Data/route_est_dat.csv
	Data/route_buffer_dat.pickle	Data/route_est_dat.pickle
	Data/route_schedule_dat.pickle	
MLPrediction	Data/route_est_dat.pickle	Results/ (all files)

A3 File Structure After Running the Scripts

```
AI Transit Prediction
├──Data
│    census_SD.pickle
│    route_buffer_dat.pickle
│    route_est_dat.csv
│    route_est_dat.pickle
│    route_schedule_dat.pickle
│    survey_2014.pickle
│
│    ├──Census_SD
│         Census_SD.cpg
│         Census_SD.dbf
│         Census_SD.prj
│         Census_SD.shp
│         Census_SD.shx
│
│    ├──GTFS
│         gtfs_schedule.pickle
│         routes.pickle
│         shapes.pickle
│         stops.pickle
│         stop_times.pickle
│         trips.pickle
│
│    └──Inputs
│         2014RouteID_ModeID.xlsx
│         NCTCOG_CONSOLIDATED_DART_THET_DCTA_Sep0315.xlsx
│
│         ├──dfw_census
│              dfw_census.cpg
│              dfw_census.dbf
│              dfw_census.prj
│              dfw_census.shp
│              dfw_census.shx
│
│         ├──GTFS
│         │    ├──DART GTFS
│                   agency.txt
│                   calendar.txt
│                   calendar_dates.txt
│                   feed_info.txt
│                   nodes.txt
│                   routes.txt
│                   route_direction.txt
│                   shapes.txt
│                   stops.txt
│                   stop_times.txt
```

```
        │       │       │       trips.txt
        │       │       │
        │       │       └──FWTA GTFS
        │       │               agency.txt
        │       │               calendar.txt
        │       │               calendar_dates.txt
        │       │               feed_info.txt
        │       │               routes.txt
        │       │               shapes.txt
        │       │               stops.txt
        │       │               stop_times.txt
        │       │               transfers.txt
        │       │               trips.txt
        │       │
        │       ├──Market Segmentation
        │       │       TAZ_Emp_IGxInd.csv
        │       │       TAZ_HH_SxWxIG.csv
        │       │       TAZ_HH_WxIGxC.csv
        │       │       TAZ_HH_WxVxIG.csv
        │       │
        │       └──TSZ5352
        │               TSZ5352.DBF
        │               TSZ5352.prj
        │               TSZ5352.shp
        │               TSZ5352.shx
        │
        ├──Notebooks
        │   │   ConsolidateTSZCensusData.ipynb
        │   │   DataPickling.ipynb
        │   │   MLPrediction.ipynb
        │   │   RouteBufferFeatures.ipynb
        │   │   RouteFeatures.ipynb
        │   │   RouteScheduleFeatures.ipynb
        │   │
        │   ├──.ipynb_checkpoints
        │   │       ConsolidateTSZCensusData-checkpoint.ipynb
        │   │       DataPickling-checkpoint.ipynb
        │   │       MLPrediction-checkpoint.ipynb
        │   │       RouteBufferFeatures-checkpoint.ipynb
        │   │       RouteFeatures-checkpoint.ipynb
        │   │       RouteScheduleFeatures-checkpoint.ipynb
        │   │
        │   └──MyLib
        │       │   LayerFuse.py
        │       │   __init__.py
        │       │
        │       ├──.ipynb_checkpoints
        │       │       LayerFuse-checkpoint.py
        │       │
```

```
        └──__pycache__
               LayerFuse.cpython-37.pyc
               __init__.cpython-37.pyc

└──Results
      AdaBoost.png
      AdaBoost_Imp.png
      DecisionTree.png
      DecisionTree_Imp.png
      DecisionTree_tree.png
      fit_summary.csv
      GradientBoost.png
      GradientBoost_Imp.png
      LassoCV.png
      LassoCVLog.png
      LassoCVSq.png
      LassoCVSqrt.png
      LassoCV_Coef.png
      Predictions.csv
      RandomForest.png
      RandomForest_Imp.png
      RidgeCV.png
      RidgeCV_Coef.png
```