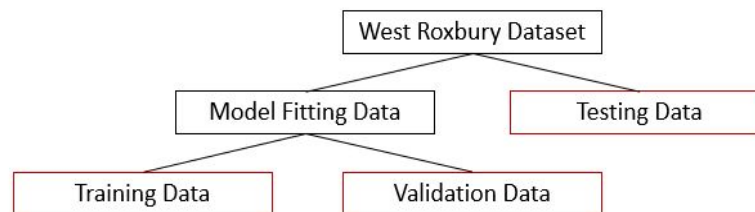


Final Project Report

By John Zizzadoro and Katelyn McGuigan

For this project, we were tasked with finding a dataset that we could manipulate and analyze using techniques we learned in MA 346. We used a dataset from GitHub containing the value, tax assessment, and features of about 5,000 homes in West Roxbury, MA (a neighborhood in Boston). The data was collected in 2018. The technique from class we chose to apply was machine learning. Our goal was to be able to predict the value of a home using some or all of its remaining data. Besides home value, the columns in the dataset were tax, lot square footage, year built, gross floor area, living area (square footage), number of floors, number of rooms, number of bedrooms, number of full bathrooms, number of half bathrooms, number of kitchens, number of fireplaces, and whether the house was remodeled recently, not recently, or never. Logically, one should be able to arrive at a property value with some degree of certainty using this data, especially since the houses are in the same neighborhood.


In order to create a machine learning model, we needed some data to fit the model to (in essence, so the “machine” could “learn”) and other data to test the model on. So, the first step we took was splitting the dataset in two. The first, larger, dataset was used for the fitting. The other dataset was used at the very end to test the model. This dataset was not opened or used until the very end. If we had looked at or used the data inside the testing dataset before we finalized our model, we may have tried to cater our model to fit that data. Gearing the model to a specific set of data could weaken the model when applied to other data. We further split the first dataset - one larger portion to fit (train) the model to, and a second, smaller portion to validate the model. Now that the data was appropriately divided (one dataset with two parts, one to train the model and one to validate and refine the model, and another dataset to run the final test on), we could move on to actually creating the model. Below is a graphic representing how the data was split. Outlined in red are the final datasets that were used at some point in the process:



We first had to decide which columns would serve as “predictors” and which column would serve as the “response.” This was fairly obvious - we wanted to predict the value of a home using the other given data. Home value was the response, and the rest of the columns were the predictors. We dropped the tax column since it was too strong of a predictor and could cause overfitting. This data would fit well to a linear model, so we used a Python package called Scikit-learn to fit the data to a linear regression using the previously stated response and

predictors. After this initial model was created, we needed to see how good it was. We used root mean square error (RMSE) to do this. This would tell us, on average for the whole dataset, how far off the model-predicted home value was from the actual home value. We took these steps on both the datasets (train and validate) within the first dataset. The RMSE values for both datasets were very similar, so this meant we had a model that was not “overfit” to the training data. An overfit model would work well on the training dataset on which the model was created, but would not work as well on other datasets. Below is a screenshot of the RMSE of the predictions for the training and validation datasets:

```
def fit_model_to ( training ):  
    predictors = training.iloc[:,2:]  
    response = training.iloc[:,0]  
    model = LinearRegression()  
    model.fit( predictors, response )  
    return model  
  
def score_model ( M, validation ):  
    predictions = M.predict( validation.iloc[:,2:] )  
    actual = validation['TOTAL_VALUE']  
    mse = ( ( predictions - actual ) ** 2 ).mean()  
    rmse = mse ** .5  
    return rmse  
  
model = fit_model_to( df_training )  
score_model( model, df_training ), score_model( model, df_validation )
```



(39.70716119590841, 40.973494561893176)

The RMSEs were both reasonable and similar, which is a sign that the model is strong. The model fits similarly to both datasets and produces a good estimate of home value (on average, about \$40,000 from the actual value for both datasets).

Next, we wanted to see if we could make the model even better. We repeated the same steps as before, but this time we dropped columns that had small absolute value coefficients in the model, meaning they were weaker predictors of the home value. Below is a screenshot of the coefficients in the original model and RMSEs on the training and validation datasets:

```
def fit_model_to ( training ) :
    # choose predictors and fit model as before
    predictors = training.iloc[:,2:]
    response = training.iloc[:,0]
    model = LinearRegression()
    model.fit( predictors, response )
    # fit another model to standardized predictors
    standardized = ( predictors - predictors.mean() ) / predictors.std()
    temp_model = LinearRegression()
    temp_model.fit( standardized, response )
    # get that model's coefficients and display them
    coeffs = pd.Series( temp_model.coef_, index=predictors.columns )
    sorted = np.abs( coeffs ).sort_values( ascending=False ) # these two lines are the
    coeffs = coeffs.loc[sorted.index] # optional bonus, sorting
    print( coeffs )
    # return the model fit to the actual predictors
    return model

# make sure it works
model = fit_model_to( df_training )
print( score_model( model, df_training ), score_model( model, df_validation ) )
```

LIVING_AREA 43.399468
 LOT_SQFT 24.388342
 FULL_BATH 13.741432
 GROSS_AREA 13.597482
 FLOORS 10.869629
 HALF_BATH 10.371919
 FIREPLACE 8.454506
 REMODEL 3.852926
 YR_BUILT 2.708906
 KITCHEN -1.694529
 ROOMS 0.484902
 BEDROOMS 0.058213
 dtype: float64
 39.70716119590841 40.973494561893176

We then tested models that dropped columns with the smallest coefficients. If the RMSE was smaller, the model was better. If not, we added back the columns we had dropped for that test. Ultimately, we decided to drop only the Bedrooms and Rooms columns. We had our final model after removing these two columns as predictors. Below is a screenshot of the coefficients for the new model, and the RMSEs it produced for the training and validation datasets:

```
columns = [0,1,2,3,4,5,6,9,10,11,12,13]
model = fit_model_to( df_training.iloc[:,columns] )
score_model( model, df_training.iloc[:,columns] ), score_model( model, df_validation.iloc[
```

```
LIVING_AREA    43.628332
LOT_SQFT       24.389517
FULL_BATH      13.792354
GROSS_AREA     13.679672
FLOORS         10.955146
HALF_BATH      10.418948
FIREPLACE      8.458562
REMODEL        3.854726
YR_BUILT        2.688962
KITCHEN        -1.646869
dtype: float64
(39.708595354563236, 40.9614222927791)
```

The RMSEs are very close to the old model and the new model. For the training dataset, the RMSEs are within \$1 of each other. For the validation dataset, the RMSEs are within \$10 of each other, with the new model having the edge.

Finally, we could run our model on the test dataset. We had not touched this dataset up until this point. Running the model on this dataset produced a RMSE smaller than that of both the test and validate datasets, indicating that our model was strong. This is an unlikely but favorable outcome that we were satisfied with. Our model was a good predictor of home value. Below is a screenshot of the RMSE of the prediction:

```
columns = [0,1,2,3,4,5,6,9,10,11,12,13]
model = fit_model_to( df_testing.iloc[:,columns] )
score_model( model, df_testing.iloc[:,columns] )
```

```
LIVING_AREA    18.372114
GROSS_AREA     16.753074
LOT_SQFT       16.075729
YR_BUILT        15.732040
FLOORS         10.898649
FULL_BATH       9.720794
REMODEL         5.338402
HALF_BATH       5.095066
KITCHEN         0.960627
FIREPLACE       0.798718
dtype: float64
31.231805168350977
```

RMSE of 31.231 indicates that on average, the model predicted home values \$31,231 off of what their actual value is.

Since we were able to create a model that predicted the home values with minimal error, we decided to create a [dashboard](#) to help people determine the value of a home if they were looking in the West Roxbury area based on certain criteria that they could choose. The dashboard allows users to input data about a home and receive an estimate of the value of that home if it were in West Roxbury, MA.

Dashboard link:

<https://sheltered-anchorage-80716.herokuapp.com/>

Repo link:

<https://github.com/katemcguigan/MA-346-Final-Project1>

Deepnote link:

<https://deepnote.com/project/bc4ffef2-9f62-4641-8933-72dfec81f866>