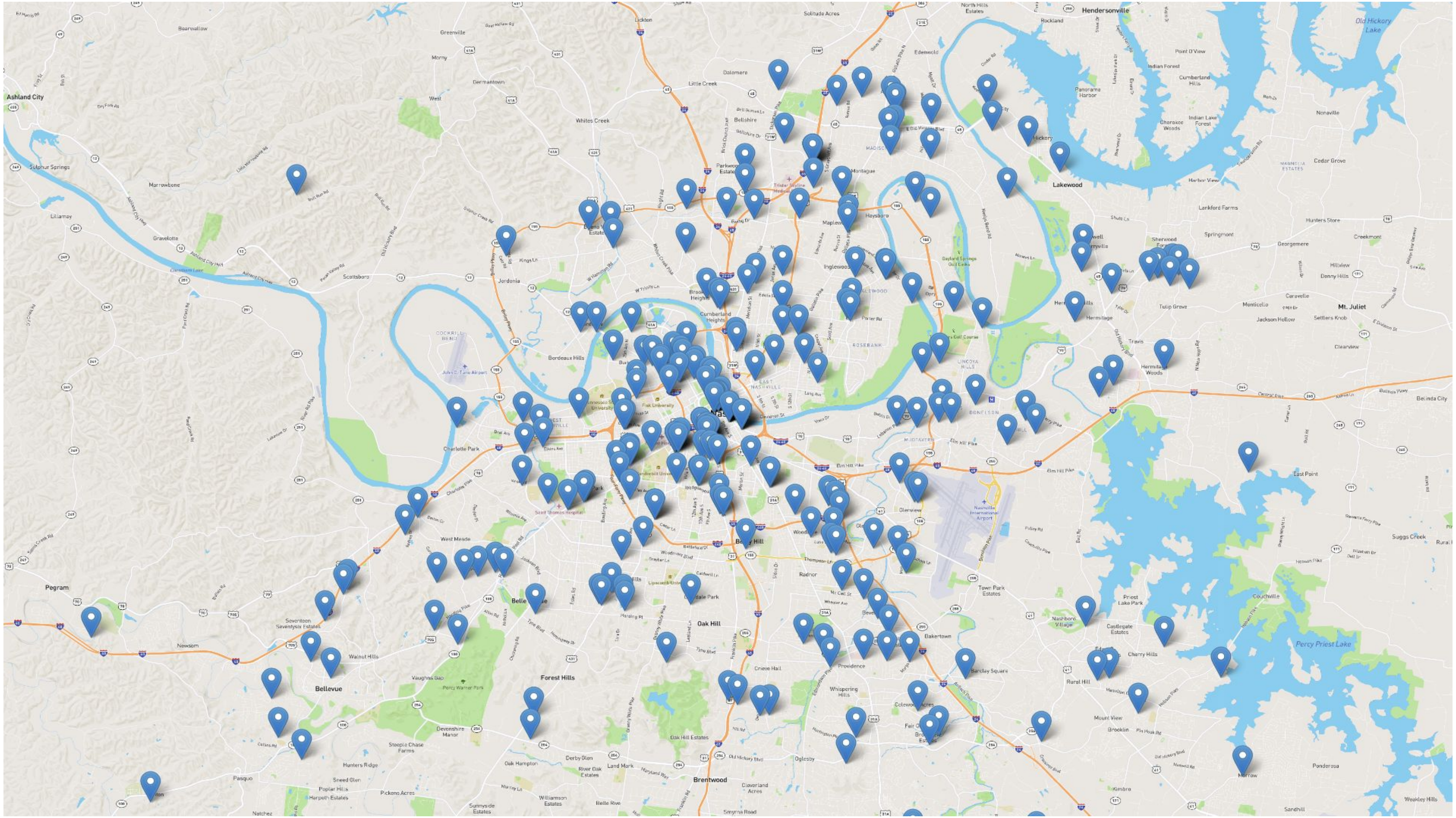
Machine Learning and Real Estate

Helping buyers find home pricing deals using Machine Learning.

Nashville, ranks in the top 10 of fastest growing metros.

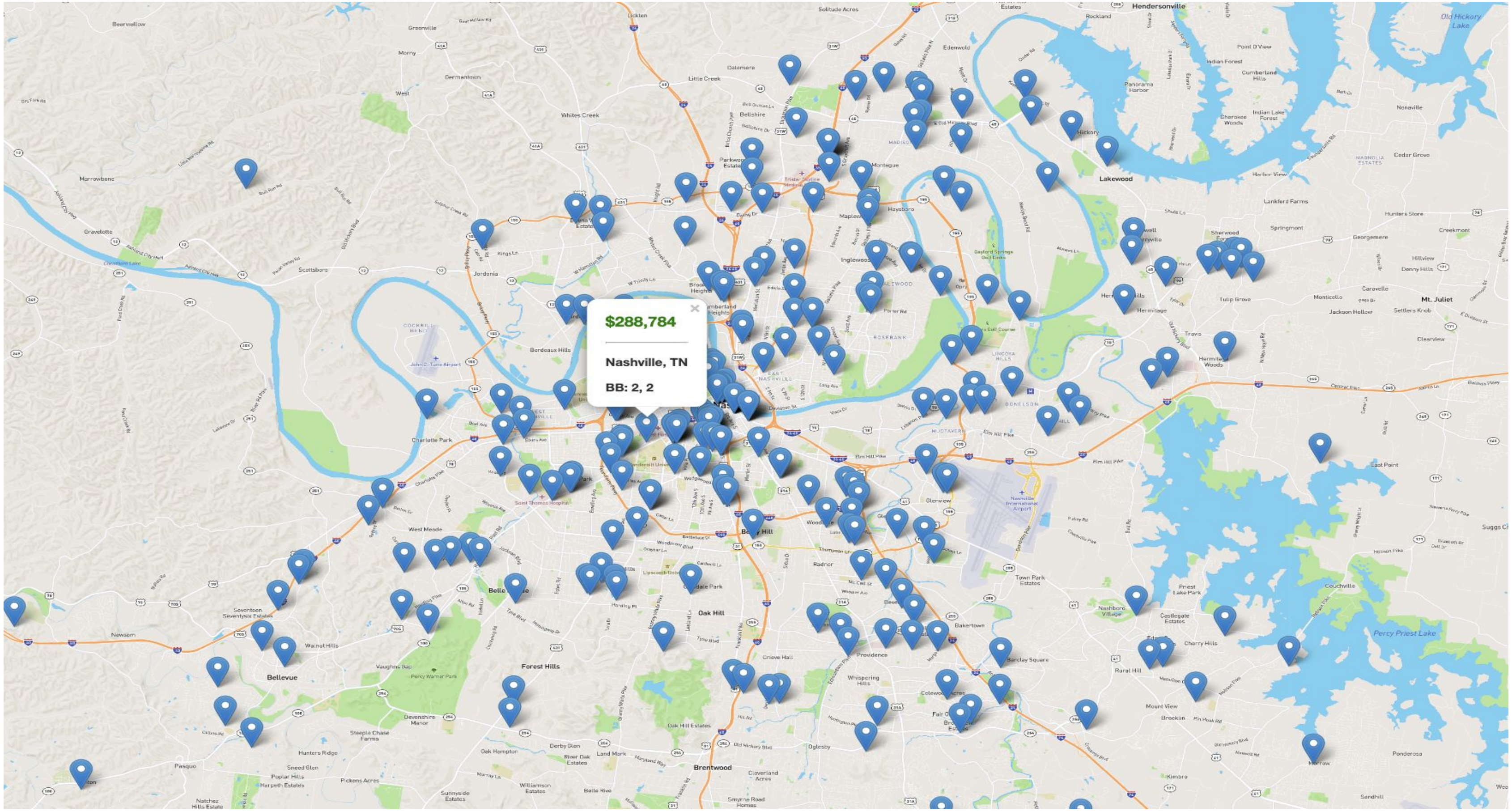
Nashville, TN and and the surrounding boroughs are predicted to be one of the top five housing markets in the country again this year. Historic low and near zero Federal Reserve rates have fueled far more buyers than sellers. With an imbalance of buyers to sellers, home prices can exceed the appraised value of the property. Additionally, many buyers exacerbate the problem by over bidding the value of the home feeding the continuation of over priced homes and non accommodating sellers.

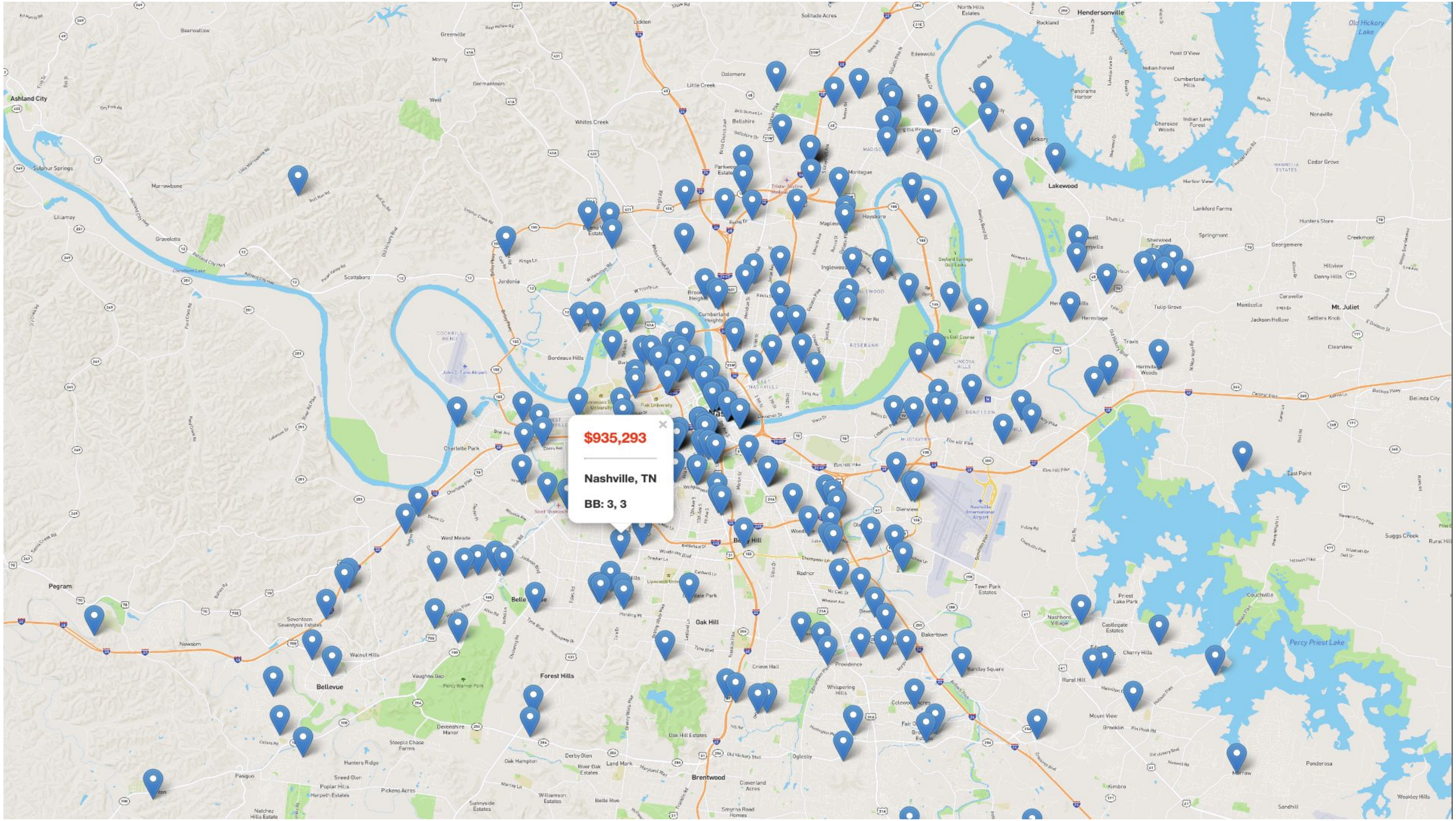
The Grey team will use Machine Learning to examine homes for sale and compare the current asking price, along with other predefined features, to previous sold homes to determine if the listing price is fair, above, or below market value.



The Problem

Many buyers are finding that the market is moving faster than they can process whether the home's asking price is a good deal (fair market value) or a bad deal (above market value). With Machine Learning, the Grey Team will process the current listings of homes for sale and develop a map of homes that are defined as good or bad value. Home buyers can click on a marker on the map and the color code of the price will alert the buyer if the home is above market value, by showing the home price in Red or fair value listing the home price in Green. Using the map provided by the Machine Learning algorithm, prospective home buyers can concentrate their efforts and resources on homes in the fair market value to make competitive bids.


$288,784
Nashville, TN
BB: 2, 2
Hendersonville
Old Hickory Lake
Lakewood
Hickory
Madison
Inglewood
Rosebank
East Nashville
Donelson
Hermitage
Mt. Juliet
Percy Priest Lake
Nashville International Airport
Antioch
Brentwood
Oak Hill
Forest Hills
Belle Meade
Bellevue
West Meade
Charlotte Park
Bordeaux Hills
Whites Creek
Joelton
Goodlettsville
Ashland City
Scottsboro
Jordonia
Cockrill Bend
John C. Tune Airport
Percy Warner Park
Vanderbilt Univ
Saint Thomas Hospital
Lipscomb Univ
Berry Hill
Woodbine
Glencliff
Radnor
Crieve Hall
Whispering Hills
Providence
Bakertown
Barclay Square
Town Park Estates
Nashboro Village
Priest Lake Park
Castlegate Estates
Couchville
Rural Hill
Mount View
Kimbro
Sandhill
Morrow
Ponderosa
Newsom
Pasquo
Pegram
Marrowbone
Bearwallow
Greenville
Morny
Germantown
West
Little Creek
Dalemere
Solitude Acres
Lickton
Edenwold
Rockland
Cedar Grove
Nonaville
Tulip Grove
Springmont
Caravelle
Monticello
Jackson Hollow
Clearview
East Point
Suggs Cr
Sunnyside Estates
Williamson Estates
Oak Hampton
Natchez Hills Estate
Harpeth Estates
Hunters Ridge
Steeple Chase Farms
Devonshire Manor
Vaughns Gap
Cloverland Acres
Smyrna Road Homes
Oglesby
Derby Glen
River Oak Estates
Land Mark
Belle Rive
Oak Hill Estates


$935,293
Nashville, TN
BB: 3, 3
Hendersonville
Old Hickory Lake
Ashland City
Lakewood
Mt. Juliet
Pegram
Bellevue
Forest Hills
Oak Hill
Brentwood
Percy Priest Lake

Machine Learning

Two data sets were needed to create our predictive model on housing price. One including recently sold houses in the Nashville metropolitan area (train and test data) and one for houses to be sold.

Finding a working API to pull sold housing data proved harder to find than initially thought. Realtors try their best to keep this data private, in an attempt to keep themselves relevant for buyers.

Without being able to find a working API, we manually exported data from the Redfin Realty website.

Cleaning the Date

- The two datasets were combined and EDA was completed.

-Unnecessary data and columns with a large number of nulls were dropped.

- Data was pushed to the The RDS Database using SQLAlchemy.

```
df.groupby(['CITY']).count()['MLS#']
```

```
CITY
Antioch             640
Ashland City          1
Brentwood           207
Franklin              3
Goodlettsville       56
Hermitage           280
Joelton              28
LA VERGNE            13
Madison             188
Mount Juliet         13
Nashville          3961
Nolensville          61
Old Hickory         142
Pegram                4
Smyrna                4
Whites Creek         21
Name: MLS#, dtype: int64
```

Database Connections and EDA

Machine Learning Testing

- Multiple models were tested (Multiple Linear Regression, Xg Boost Regression, and Neural Network). Absolute Mean Error, Mean Squared Error and Root Mean Squared Error were used to measure loss and predicted accuracy on the test data.
- Neural network performed the best in all metrics and was selected. Different variations of features were used to find the best measures.
- Hyper parameter testing helped select kernel initializer and activation function using Sklearn's GridSearchCV.
- Code was run against multiple different numbers of epochs to confirm the best model accuracy.

Neural Network

```
#Define the model - deep neural net
number_input_features = len(X_train[0])

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=50, input_dim=number_input_features, activation="linear")
)
nn.add(
    tf.keras.layers.Dense(units=40, activation="relu")
)

nn.add(
    tf.keras.layers.Dense(units=30, activation="linear")
)
nn.add(
    tf.keras.layers.Dense(units=20, activation="linear")
)

nn.add(
    tf.keras.layers.Dense(units=10, activation="linear")
)
# Second hidden layer
nn.add(tf.keras.layers.Dense(units=8, kernel_initializer='normal'))


# Output layer
nn.add(tf.keras.layers.Dense(units=1))

# Check the structure of the model
nn.summary()
```

Final Loss and Accuracy

```
Epoch 98/100
132/132 [==============================] - 0s 2ms/step - loss: 87315912.0000 - root_mean_squared_error: 9344.2988
Epoch 99/100
132/132 [==============================] - 0s 2ms/step - loss: 175968912.0000 - root_mean_squared_error: 13265.3271
Epoch 100/100
132/132 [==============================] - 0s 2ms/step - loss: 107268016.0000 - root_mean_squared_error: 10357.0273
```

```
prediction = nn.predict(X_test)
pred = pd.DataFrame({ 'actual': y_test})
pred['prediction'] = prediction
pred['error'] =pred.prediction - pred.actual
pred['error_abs'] = abs(pred['error'])
print(pred.error_abs.mean())
print(mean_squared_error(y_test, prediction))
print(mean_squared_error(y_test, prediction, squared=False))
```

```
10140.562294407895
928830839.9858888
30476.7262019051
```

URL for Heroku