



University of Nevada, Reno ---- Big Data

Zillow's Home Value Prediction

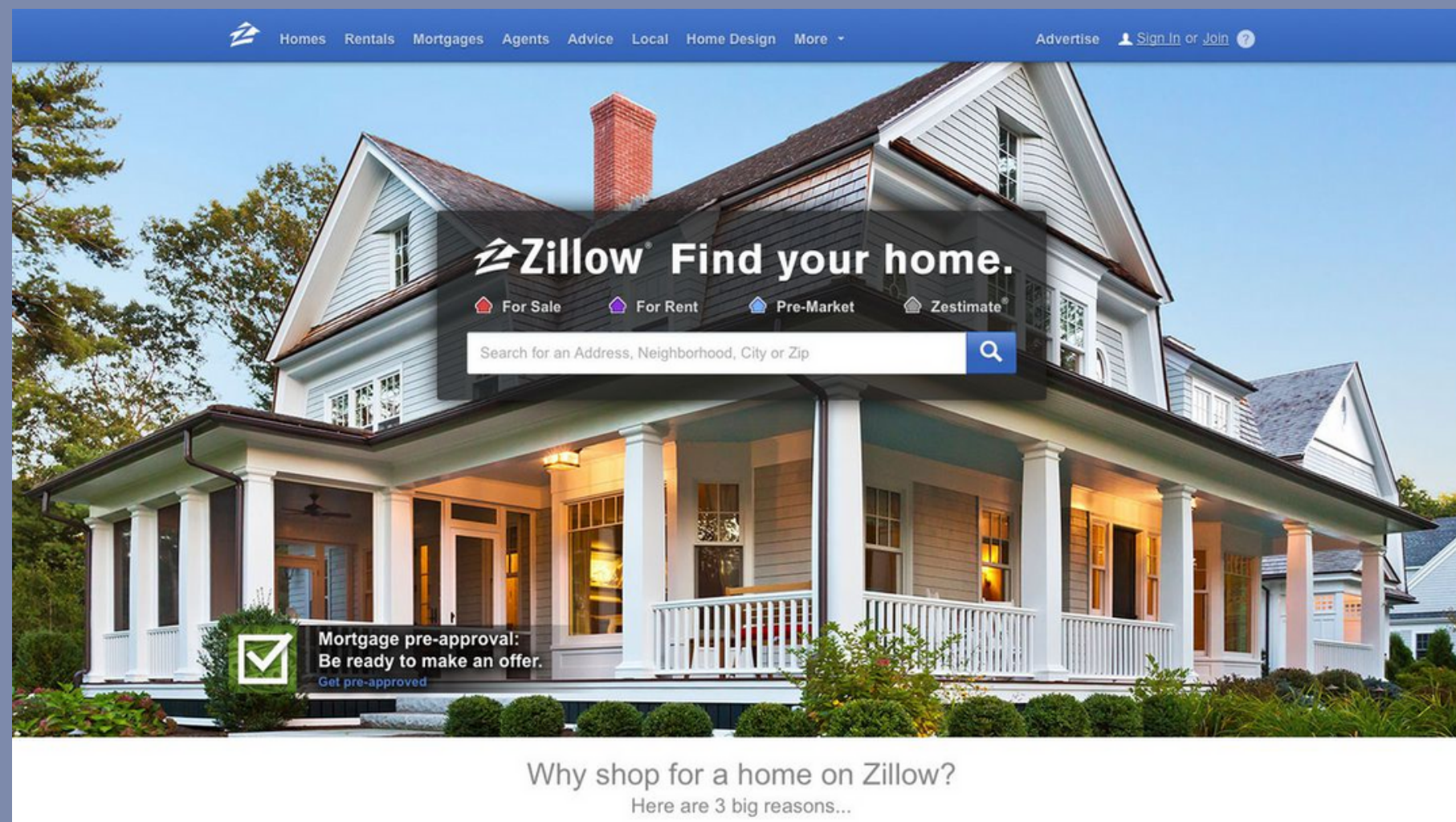


Meet the Group

Andrew Wiltberger

Zhuqi You

Yan Shore



Zillow

- Zillow is the largest full life cycle real estate marketplace
- 240 million monthly users
- Zillow also provides a Zestimates which gives users an estimation of their home values using machine learning techniques

Motivations

- With increases in housing prices accurate pricing will be more and more important.
- Zillow competitions.
- In the US the average family income is around \$60,000 a year meaning budgets for housing will be tight and accurate pricing will be important.
- People consider a mix of safety and price.

Zillow's Home Value Prediction

$$\text{logerror} = \log(\text{Zestimate}) - \log(\text{SalePrice})$$

- Given the features of a home predict the log-error between their Zestimate and the actual sale price
- Zillow offered \$1.2 Million in prize money for solutions to this problem



Tools and Technology Used

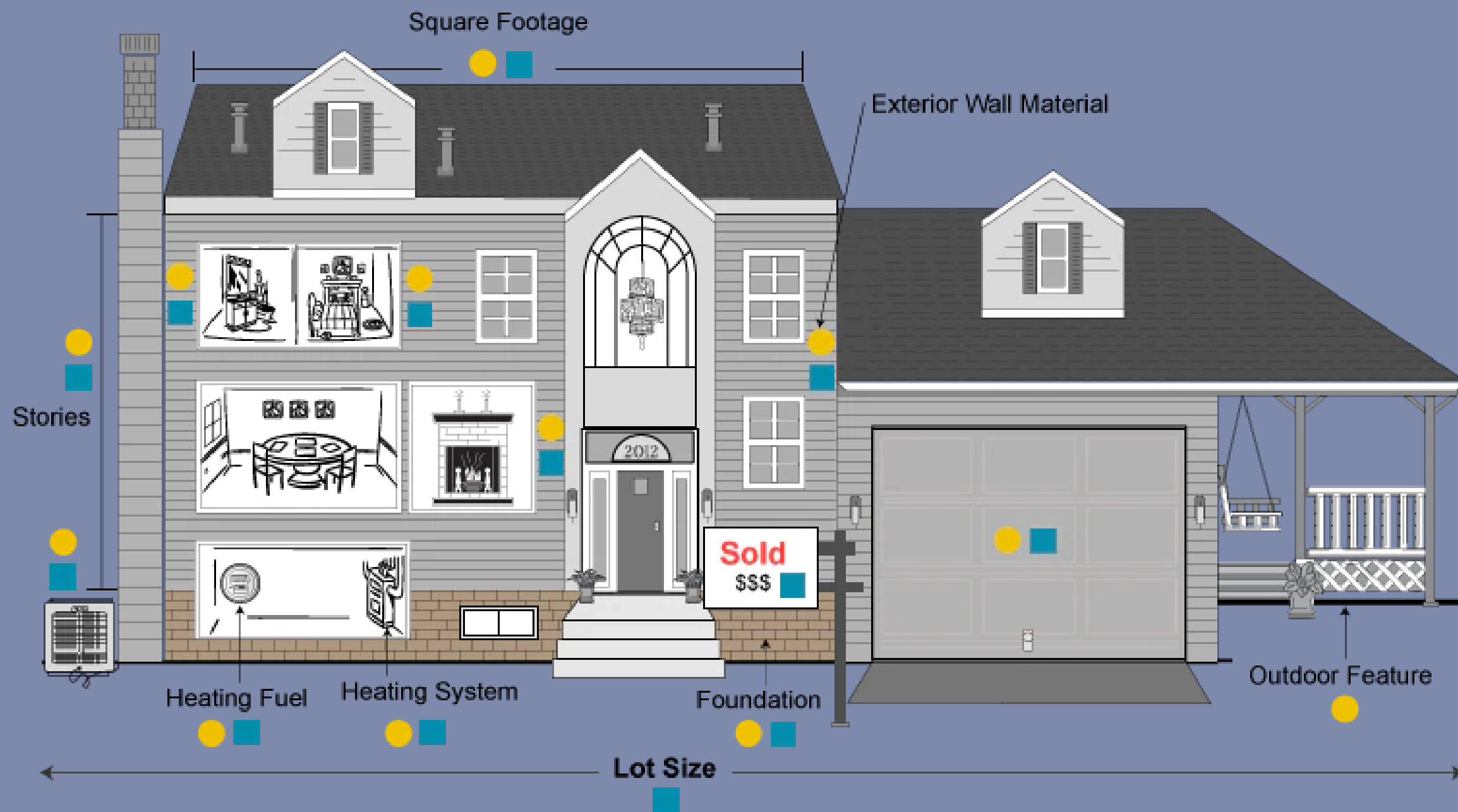


- python
- numpy
- pandas
- matplotlib
- sklearn
- seaborn
- keras



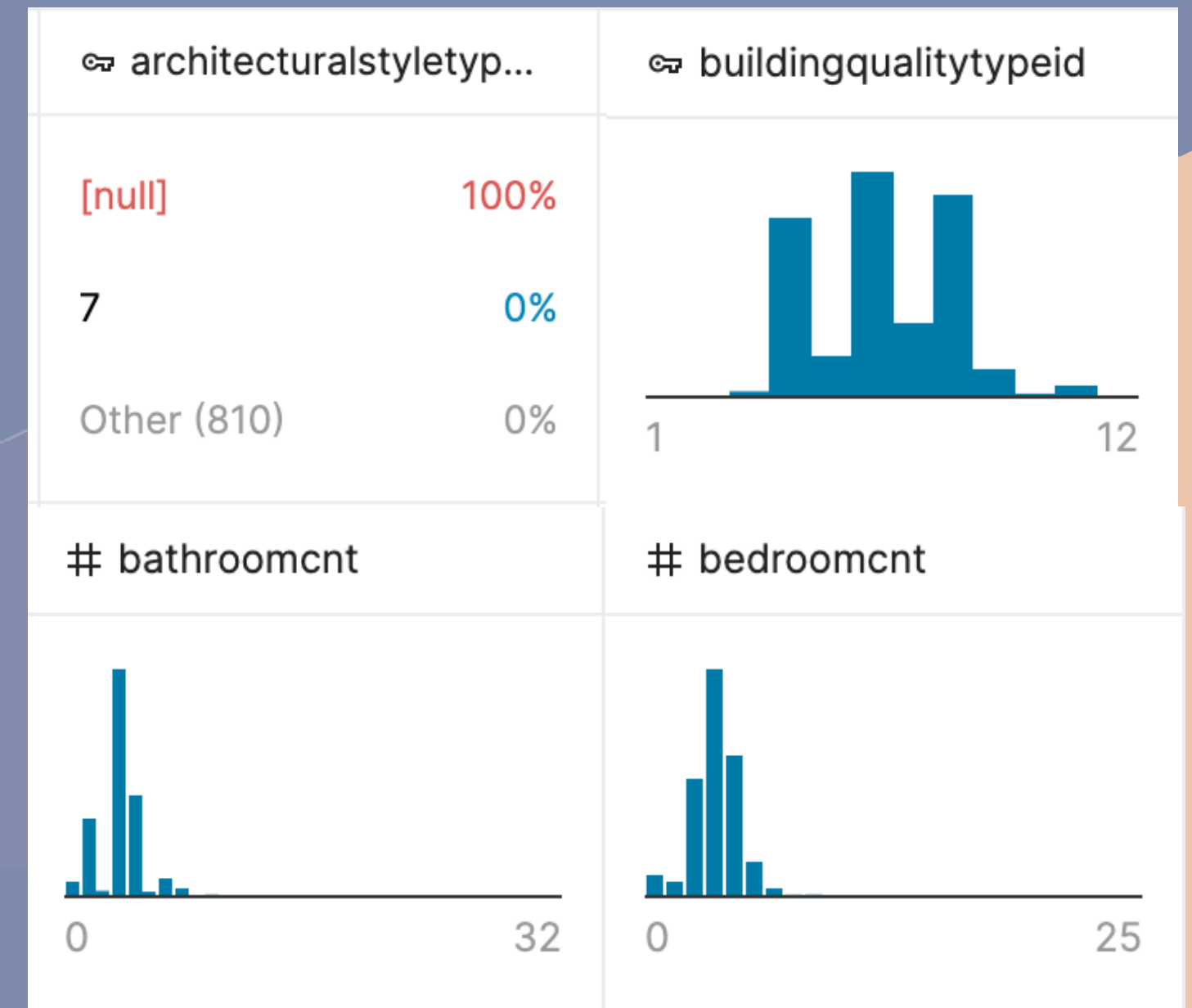
The Data

- Provided with a full list of real estate properties in three counties (Los Angeles, Orange and Ventura, California) data in 2016.
- Housing features includes existence of air conditioning, room count, square footage of rooms, if there is a pool, and many more.



Data sets

- 2016
 - 58 features
 - 100,000 + rows
- 2017
 - 58 features
 - 100,000 + rows



Cleaning data

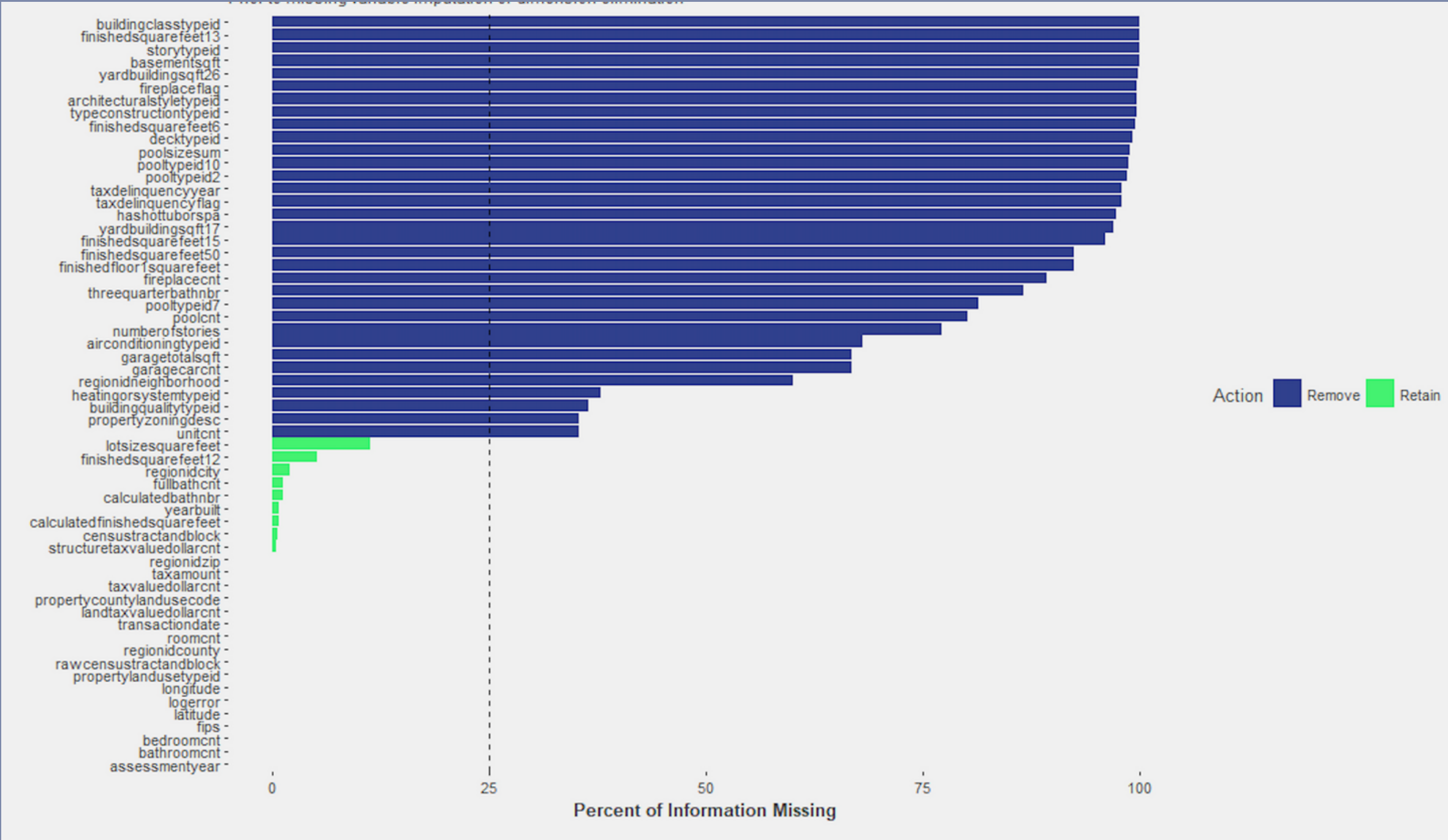
- choose attributes
 - which ones are important
 - if choose different attributes, get different results
- missing values
 - fill the missing values
 - or delete the entry

fireplace,
swimming pool...

year of build,
property tax...



missing values





Volume

Relatively low, data could be processed on mid tier laptops



Velocity

low, static data on disk



Variety

All data is in the same schema



Veracity

Some missing fields



Value

High value data that could provide useful insight to many home buyers

The data 5 V's

Solution Implementation

Step 1:

Using pandas we read the files
containing data

```
## Step 1: Import Data
```

```
df_train_16 = pd.read_csv('train_2016_v2.csv')  
df_train_17 = pd.read_csv('train_2017.csv')  
df_prop_16 = pd.read_csv("properties_2016.csv")  
df_prop_17 = pd.read_csv("properties_2017.csv")  
samplesub = pd.read_csv("sample_submission.csv")
```

Step 2:

Using pandas we merge that data for predictions purposes.

```
# Step 2: Data Merging for the prediction
data_16 = pd.merge(df_prop_16,df_train_16)
data_17 = pd.merge(df_prop_17,df_train_17)
df = pd.concat([data_16,data_17],keys=('parcelid','transactiondate'))
num_cols = [col for col in df.columns if (df[col].dtype in ['float64','int64'] and col not in ['parcelid',
temp_df = df[num_cols]
for c in df.columns:
    df[c]=df[c].fillna(-1)
    if df[c].dtype == 'object':
        lbl = LabelEncoder()
        lbl.fit(list(df[c].values))
        df[c] = lbl.transform(list(df[c].values))
```


Step 3:

Split data into training and testing sets

```
df = df.fillna(-1.0)
x_train = df.drop(['parcelid', 'logerror', 'transactiondate', 'propertyzoningdesc', 'propertycountylandusecode', 'fireplacecnt', 'fire
y_train = df["logerror"]

y_mean = np.mean(y_train)
print(x_train.shape, y_train.shape)
train_columns = x_train.columns

for c in x_train.dtypes[x_train.dtypes == object].index.values:
    x_train[c] = (x_train[c] == True)
samplesub['parcelid'] = samplesub['ParcelId']
df_test = samplesub.merge(df_prop_16, on='parcelid', how='left')
df_test["transactiondate"] = pd.to_datetime('2016-11-15') # placeholder value for preliminary version
df_test["transactiondate_year"] = df_test["transactiondate"].dt.year
df_test["transactiondate_month"] = df_test["transactiondate"].dt.month
df_test["transactiondate_quarter"] = df_test["transactiondate"].dt.quarter
```

Step 3 Continued:

Split data into training and testing sets

```
df_test["transactiondate"] = df_test["transactiondate"].dt.day
x_test = df_test[train_columns]

for c in x_test.dtypes[x_test.dtypes == object].index.values:
    x_test[c] = (x_test[c] == True)
```

Step 4:

Apply machine learning model to data

```
imputer= Imputer()
imputer.fit(x_train.iloc[:, :])
x_train = imputer.transform(x_train.iloc[:, :])
imputer.fit(x_test.iloc[:, :])
x_test = imputer.transform(x_test.iloc[:, :])

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

len_x=int(x_train.shape[1])
print(len_x)
nn = Sequential()
nn.add(Dense(units = 400 , kernel_initializer = 'normal', input_dim = len_x))
nn.add(PReLU())
nn.add(Dropout(.4))
```

Step 4 Continued(1):

Apply machine learning model to data

```
nn.add(Dense(units = 160 , kernel_initializer = 'normal'))
nn.add(PReLU())
nn.add(BatchNormalization())
nn.add(Dropout(.6))
nn.add(Dense(units = 64 , kernel_initializer = 'normal'))
nn.add(PReLU())
nn.add(BatchNormalization())
nn.add(Dropout(.6))
nn.add(Dense(units = 64 , kernel_initializer = 'normal'))
nn.add(PReLU())
nn.add(BatchNormalization())
nn.add(Dropout(.5))
nn.add(Dense(units = 26, kernel_initializer = 'normal'))
nn.add(PReLU())
nn.add(BatchNormalization())
nn.add(Dropout(.6))
```

Step 4 Continued(2):

Apply machine learning model to data

```
nn.add(Dense(1, kernel_initializer='normal'))
nn.compile(loss='mae', optimizer=Adam(lr=4e-3, decay=1e-4))
nn.fit(np.array(x_train), np.array(y_train), batch_size = 1000, epochs = 20, verbose=2)
y_pred_ann = nn.predict(x_test)
nn_pred = y_pred_ann.flatten()

pd.DataFrame(nn_pred).head()
y_pred=[]

for i,predict in enumerate(nn_pred):
    y_pred.append(str(round(predict,4)))
y_pred=np.array(y_pred)

output = pd.DataFrame({'ParcelId': df_prop_16['parcelid'].astype(np.int32),
                        '201610': y_pred, '201611': y_pred, '201612': y_pred,
                        '201710': y_pred, '201711': y_pred, '201712': y_pred})
```

Step 5:

Output results

```
# set col 'ParceID' to first col
cols = output.columns.tolist()
cols = cols[-1:] + cols[:-1]
output = output[cols]
# Output
print('Output of the model')
print('Output')
```


Results

Epoch 1/20

- 9s - loss: 0.0798

Epoch 2/20

- 9s - loss: 0.0698

Epoch 3/20

- 8s - loss: 0.0698

Epoch 4/20

- 8s - loss: 0.0695

Epoch 5/20

- 8s - loss: 0.0695

Epoch 6/20

- 9s - loss: 0.0693

Epoch 7/20

- 8s - loss: 0.0692

Epoch 8/20

- 9s - loss: 0.0690

Epoch 9/20

- 8s - loss: 0.0690

Epoch 10/20

- 8s - loss: 0.0689

Results Continued (1)

Epoch 11/20

- 8s - loss: 0.0689

Epoch 12/20

- 8s - loss: 0.0690

Epoch 13/20

- 8s - loss: 0.0688

Epoch 14/20

Epoch 15/20

- 8s - loss: 0.0688

Epoch 16/20

- 8s - loss: 0.0687

Epoch 17/20

- 8s - loss: 0.0687

Epoch 18/20

- 8s - loss: 0.0687

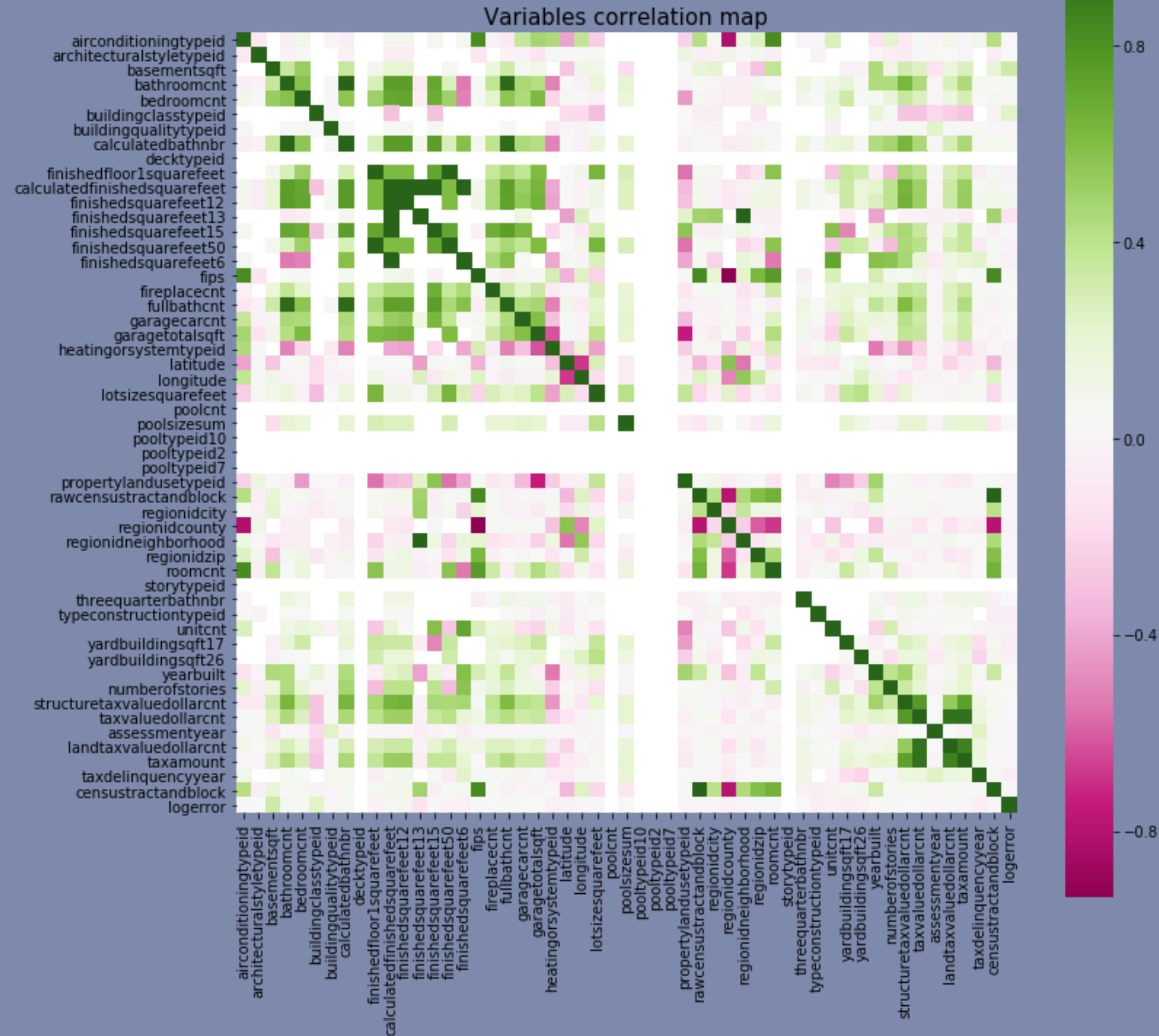
Epoch 19/20

- 8s - loss: 0.0686

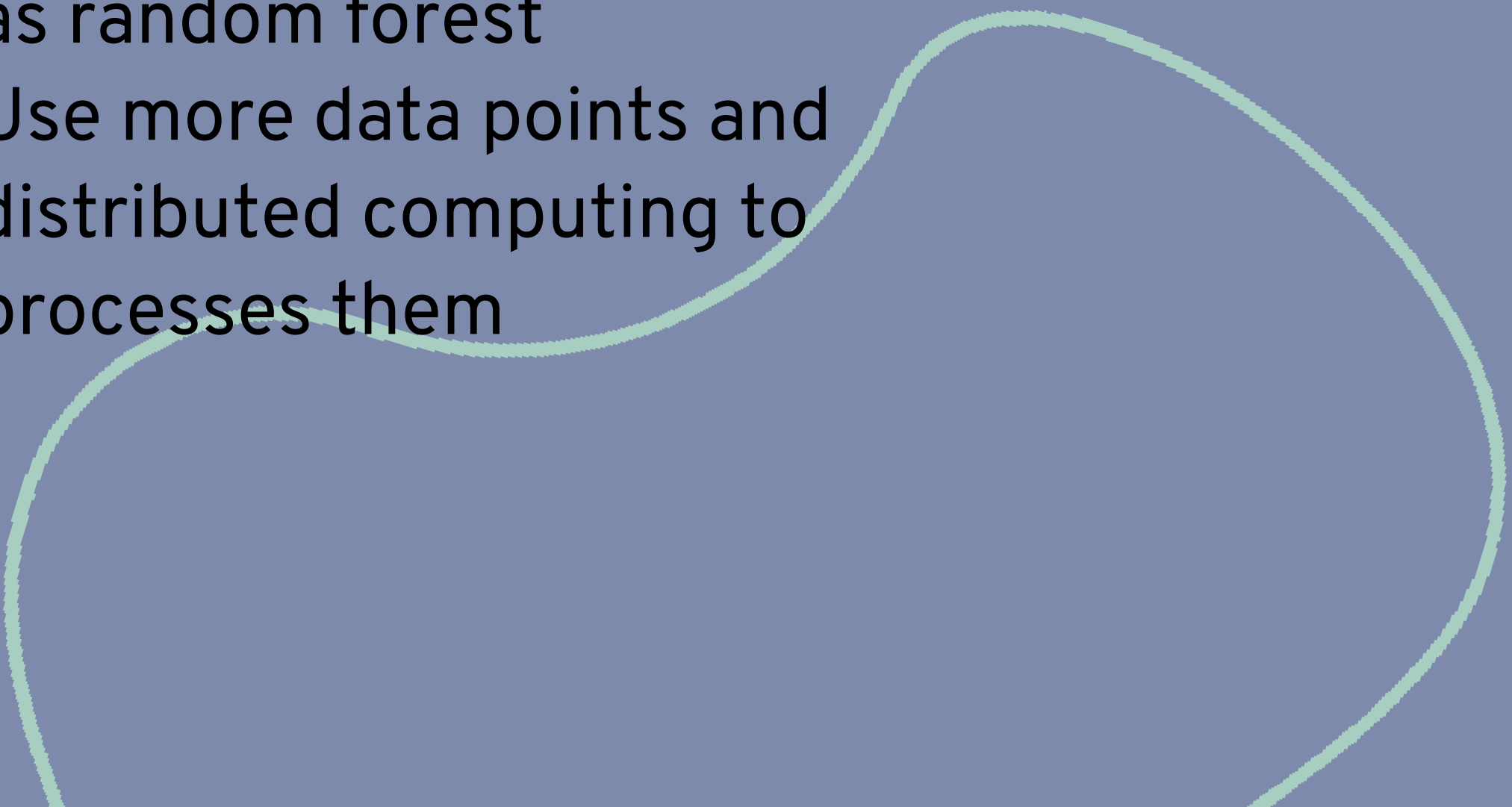
Epoch 20/20

- 8s - loss: 0.0686

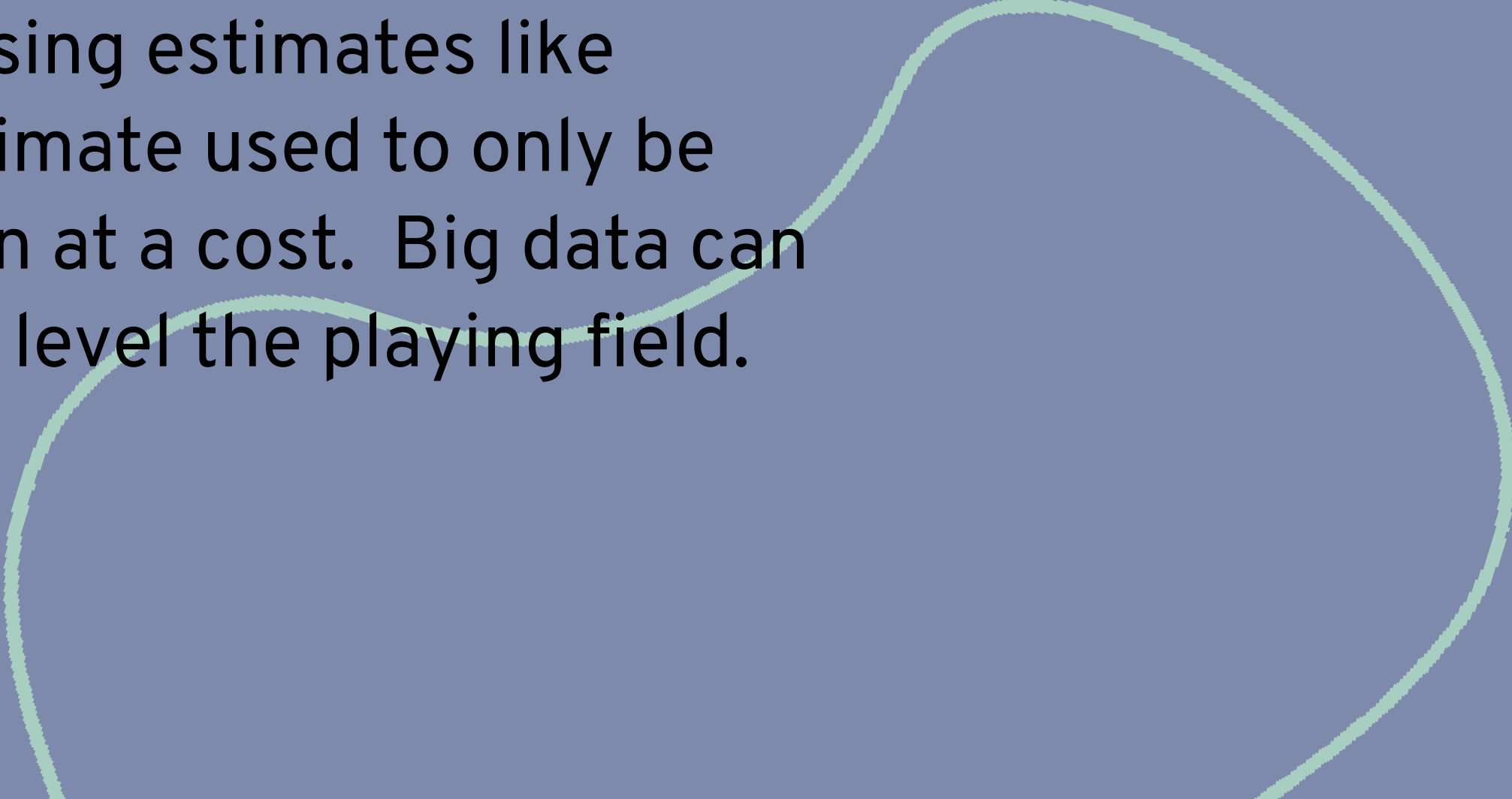
Results - Variable Correlation matrix



Future work

- Add more relevant data fields such as crime rates and local school ratings.
 - Uses other techniques such as random forest
 - Use more data points and distributed computing to processes them
- 

Conclusion

- Big data is a powerful tool that can benefit many parts of our life.
 - Can help us make informed housing choices.
 - Housing estimates like zestimate used to only be given at a cost. Big data can help level the playing field.
- 



Contributions

Zhuqi You

Implementation and presentation

Andrew Wiltberger

Presentation and implementation

Yan Shore

Presentation and implementation



Thank you!

Do you have any questions for us?