**Research Overview**

- **Problem Statement:** Build a model to predict the selling price of a condo based on 48 features pulled or derived from a dataset of downtown Vancouver condo listings under $2.5M between 2016 and 2018.
- **Data:** Pulled from an official database that contains all real estate listings in Canada, with the help of a local real estate agent:
  - Contains approximately 10,000 condo listings
  - Contains structured data (ex: square footage)
  - Contains semi-structured data (ex: address)
  - Contains unstructured data (ex: “has view?”)

**Data Exploration**

- Used Google Maps API to map condo addresses to (lat, lon) coordinates and further mapped to a rectangular grid to approximate neighborhood.
- Total sales area is ~9 km².
- Top features highly correlated with **Sold Price**:
  - Number of bedrooms
  - Number of bathrooms
  - Total Floor Area
  - Number of Parking Spaces
  - Maintenance Fees
  - Gross Taxes

**Feature Engineering**

- 48 features modelled
- NLP used to create “view score” feature
- Categorical features were one-hot encoded
- Numerical features normalized (mean + standard deviation)
- Missing data in some features was imputed

**Results**

<table>
<thead>
<tr>
<th>Model</th>
<th>R² (train)</th>
<th>R² (CV)</th>
<th>MSE (train)</th>
<th>MSE (CV)</th>
<th>MSE (CV) / MSE (train)</th>
<th>R² (text)</th>
<th>MSE (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression</td>
<td>0.8154</td>
<td>0.8066</td>
<td>0.1839</td>
<td>0.1922</td>
<td>1.0452</td>
<td>0.8032</td>
<td>0.1998</td>
</tr>
<tr>
<td>Lasso + Linear Regression</td>
<td>0.8136</td>
<td>0.8033</td>
<td>0.1857</td>
<td>0.1955</td>
<td>1.0529</td>
<td>0.8023</td>
<td>0.2006</td>
</tr>
<tr>
<td>Ridge + Linear Regression</td>
<td>0.8150</td>
<td>0.8066</td>
<td>0.1843</td>
<td>0.1922</td>
<td>1.0426</td>
<td>0.8029</td>
<td>0.2000</td>
</tr>
<tr>
<td>Regression Tree</td>
<td>0.9512</td>
<td>0.8584</td>
<td>0.0486</td>
<td>0.1420</td>
<td>2.9212</td>
<td>0.8587</td>
<td>0.1434</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.9554</td>
<td>0.8950</td>
<td>0.0444</td>
<td>0.1042</td>
<td>2.3442</td>
<td>0.9033</td>
<td>0.0981</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td><strong>0.9200</strong></td>
<td><strong>0.8993</strong></td>
<td><strong>0.0797</strong></td>
<td><strong>0.1000</strong></td>
<td><strong>1.2545</strong></td>
<td><strong>0.9069</strong></td>
<td><strong>0.0945</strong></td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.8470</td>
<td>0.8278</td>
<td>0.1525</td>
<td>0.1750</td>
<td>1.1480</td>
<td>0.8325</td>
<td>0.1699</td>
</tr>
</tbody>
</table>

**Model: Gradient Boosting Implementation**

\[
\begin{align*}
\text{Initiate } f_0(x) &= \arg \min_{f_0} \sum_{i=1}^N L(y_i, f_0) \\
\text{For } m = 1, \ldots, M: \\
\quad \tau_m &= \text{threshold for } f_{m-1} \\
\quad \text{Fit tree to } R_{m-1}/(m-1) \\
\quad \text{For } j = 1, \ldots, J_m: \\
\quad \quad \tau_m^j &= \arg \min_{\tau_m^j} \sum_{i=1}^{N_m^j} L(y_i, f_{m-1} + \tau_m^j x_i) + \rho \\
\quad \text{Output } f_m(x) &= f_{m-1}(x) + \sum_{j=1}^{J_m} \tau_m^j I(x_i \in R_m) \\
\end{align*}
\]

- Hyper-parameters with 100 estimators and max tree depth 8 produced best results in accuracy/generation trade-off
- Feature subset selected by LASSO gained no improvement
- Hyper-parameter optimization search improved CV error but rendered variance unacceptable
- Doubling estimator count and restricting max number of features used at split to $\log_2$ features gave error and variance estimates close to default

**Model: Neural Network Implementation**

- Experimented with various layer depths + sizes, and activation functions. In general deeper networks with smaller layers performed better.
- Overfitting was initially very high (test dataset had 10x MSE of train). Use of a dropout function after first layer, and running a 5-Fold CV reduced difference to less than 15%

**Discussion & Conclusions**

- **Linear Regression:** the large observed bias is due to limitations of linear method
- **Regularization** with LASSO & Ridge only marginally improved the error and made us try nonlinear learning methods
- **Regression Trees & Random Forest:** overfitting was addressed with decreasing maximum tree depth and lifted prediction accuracy to a new level
- **Neural Networks** with optimal configuration gave prediction accuracy similar to the regression tree
- **Gradient Boosting Regression** finally provided most robust and accurate model leveraging non-linear nature of interaction between features and target variable

**Fun Facts:**

- As the plot shows, the more expensive the condo, the more subjective the price appears to be (and the worse our model performs)
- **Listing price,** the quintessence of a real estate agent’s domain knowledge about current market situation, outperforms our best prediction with a 6x lower MSE

**Future Work**

There is more work to be done around modeling the temporal aspect of condo prices. Further feature engineering (30 day sale volume) and data collection (new condo developments under construction) would likely further improve our model’s accuracy.

**References**