PC game play time estimation based on Steam data and reviews

Summary
The Steam platform is the largest PC game online distributor in the world, and has accumulated a vast amount of player and game data.

In this project, our goal is to predict the average total playtime of a game based on its metadata (e.g. genre, price and publisher) and reviews (votes up and review text). Our best model was a random forest using the mean absolute error as criterion and logarithm of the playtime as output feature. This model was able to predict the average total playtime to within an error of 50% for 62.8% of all the games in our test set. Although certain 2-grams and 3-grams were among the most informative features, adding text-based features did not improve the overall performance significantly.

Features

<table>
<thead>
<tr>
<th>Feature (256 columns)</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>one-hot encoded with an 'others' column</td>
</tr>
<tr>
<td>publisher</td>
<td>raw input (unscaled)</td>
</tr>
<tr>
<td>genre</td>
<td>weighted frequency</td>
</tr>
<tr>
<td>initial price</td>
<td>weighted frequency</td>
</tr>
<tr>
<td>number of up-votes</td>
<td>weighted frequency</td>
</tr>
<tr>
<td>up- to down-votes ratio</td>
<td>weighted frequency</td>
</tr>
<tr>
<td>number of owners</td>
<td>weighted frequency</td>
</tr>
<tr>
<td>achievements to unlock</td>
<td>weighted frequency</td>
</tr>
<tr>
<td>review text features*</td>
<td>weighted frequency</td>
</tr>
</tbody>
</table>

* Text features

• After tokenizing the sentences, lemmatizing and removing stop words in there, we hand-picked 62 popular 1-grams, 2-grams and 3-grams that we thought indicate long playtime.

Moreover, we divided by total count of 1-, 2- and 3- grams in these n-grams.

• Text features would be included as a column in features

Results

Results - Experimenting with Models and Features

<table>
<thead>
<tr>
<th>Model</th>
<th>Games with more than 10% error</th>
<th>Games with more than 50% error</th>
<th>Games with more than 90% error</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predicting average</td>
<td>65%</td>
<td>47%</td>
<td>80%</td>
<td>75%</td>
<td>87%</td>
</tr>
<tr>
<td>Predicting median</td>
<td>18%</td>
<td>19%</td>
<td>47%</td>
<td>48%</td>
<td>62%</td>
</tr>
<tr>
<td>RF (MSE)</td>
<td>26%</td>
<td>22%</td>
<td>48%</td>
<td>38%</td>
<td>55%</td>
</tr>
<tr>
<td>RF (MAE)</td>
<td>18%</td>
<td>14%</td>
<td>40%</td>
<td>29%</td>
<td>58%</td>
</tr>
<tr>
<td>RF (MSE), log(y)</td>
<td>13%</td>
<td>9%</td>
<td>38%</td>
<td>26%</td>
<td>57%</td>
</tr>
<tr>
<td>RF (MAE), log(y)</td>
<td>12%</td>
<td>9%</td>
<td>37%</td>
<td>27%</td>
<td>56%</td>
</tr>
<tr>
<td>GB (naive sq, log(y))</td>
<td>14%</td>
<td>15%</td>
<td>40%</td>
<td>59%</td>
<td>59%</td>
</tr>
</tbody>
</table>

• Because the distribution of playtime is skewed, we decided to use log(playtime) as output feature and the mean absolute loss to prevent few games with very long playtime to dominate the cost function.

• On the right, we present the result in terms of percentage of games with less than 5%, 10%, 25%, 50% and 90% error.