MATH 156 Final Project

Predicting videogame sales with various models

Group 5

University of California, Los Angeles

July 27, 2020

Table of Contents

Table of Contents

K Nearest Neighbor Regression

Goal: given $x \in \mathbb{R}^d$, predict sales.

- Find k nearest data points to x.
- Compute the predicted sales based on these k points.

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=k).fit(X_train,
      Y_train)
res = model.predict(X_test, Y_test)
```

K Nearest Neighbor Regression

Goal: given $x \in \mathbb{R}^d$, predict sales.

- Find k nearest data points to x.
- Compute the predicted sales based on these k points.

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=k).fit(X_train,
      Y_train)
res = model.predict(X_test, Y_test)
```

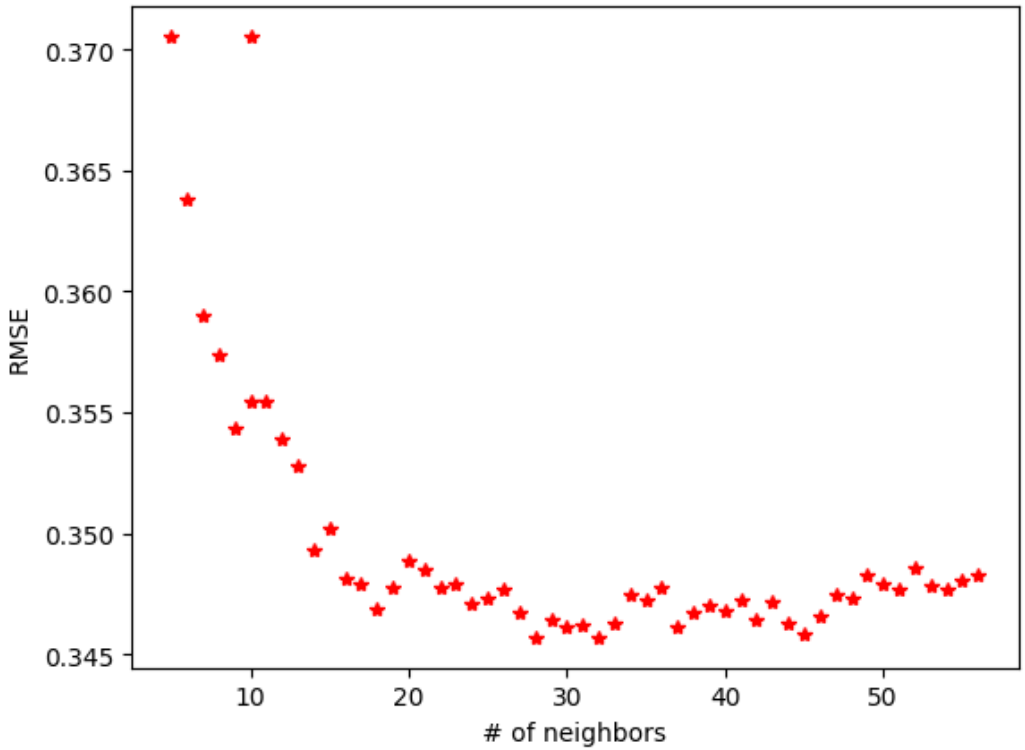
Questions we should think about:

- How to determine k?
- How to find the nearest points efficiently?
- How to predict the sales based on the points?

Cross Validation

How to determine k?

- Divide the training dataset into two parts $*$
- K-fold cross validation: divide the data into p equal parts
  - $\epsilon_p(k) = \sum_{i \in p\text{th part}} (y_i - f(x_i))^2$
  - $\epsilon(k) = \frac{1}{p} \sum_{i=1}^{p} \epsilon_p(k)$



Kd-Tree

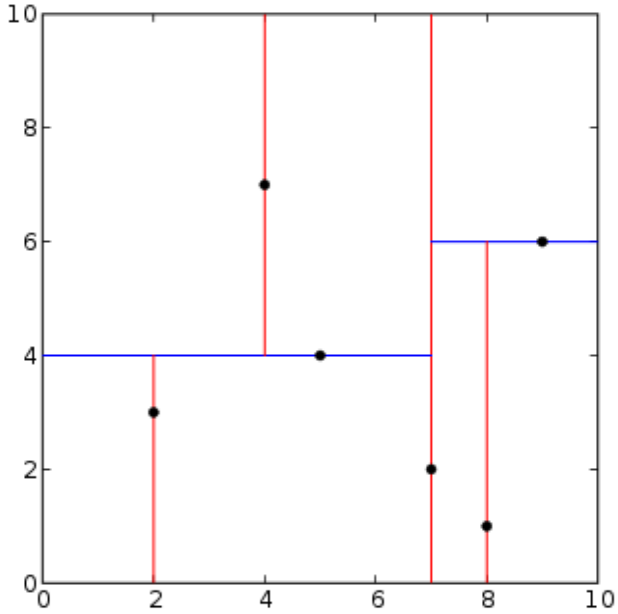
How to find the nearest points efficiently given that the training size is n with dimension d, assuming that we are using the Euclidean metric?

- Naive approach
  - Compare with all data points in the training set
  - Time Complexity: $\Theta(nd)$

Kd-Tree

How to find the nearest points efficiently given that the training size is n with dimension d, assuming that we are using the Euclidean metric?

- Naive approach
  - Compare with all data points in the training set
  - Time Complexity: $\Theta(nd)$
- Kd-Tree
  - Construct a Kd-Tree by recursively partition the plane to two halves and balancing it.
  - Search in a bushy binary tree
  - The number of features we use is small.
  - Time Complexity: $\Theta(d \log n)$.



Sales Prediction

Given $(x_1, y_1), \ldots, (x_k, y_k)$, and x, how should we predict sales based on nearest points?

- Weighted Mean
  - $d_{\text{total}} = \sum_{i=1}^{k} d(x, x_i)$
  - $y = \sum_{i=1}^{k} \frac{d(x,x_i)y_i}{d_{\text{total}}}$
- Median
- Linear Regression

Results

For our purposes, we used the K Nearest Neighbors Regressor from sklearn with Kd-Tree and weighted distance for prediction. (RMSE: 0.33)

- Pros
  - No assumptions about the data
- Cons
  - Localized data when k increases ($k = 20$ in this case).
  - Memory inefficient and slow

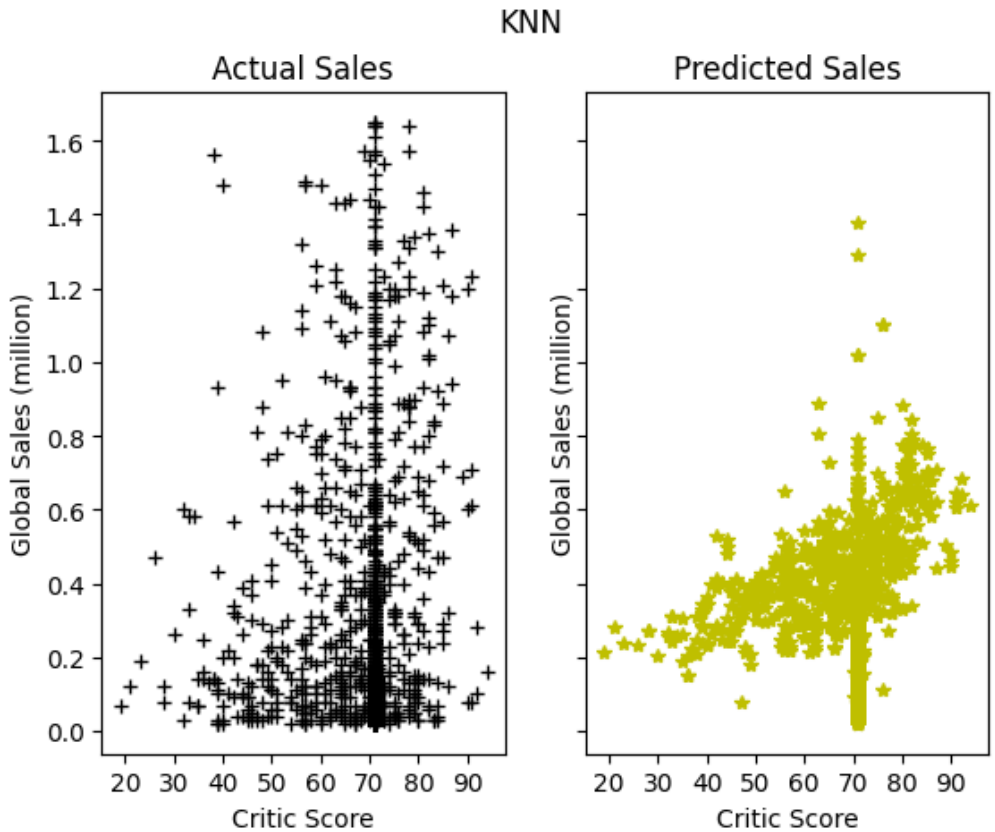


Table of Contents

Random Forest

- The Random Forest model is known as an ensemble learning method which uses multiple decision trees to be trained and in the case of regression, outputs the mean prediction of the individual trees.

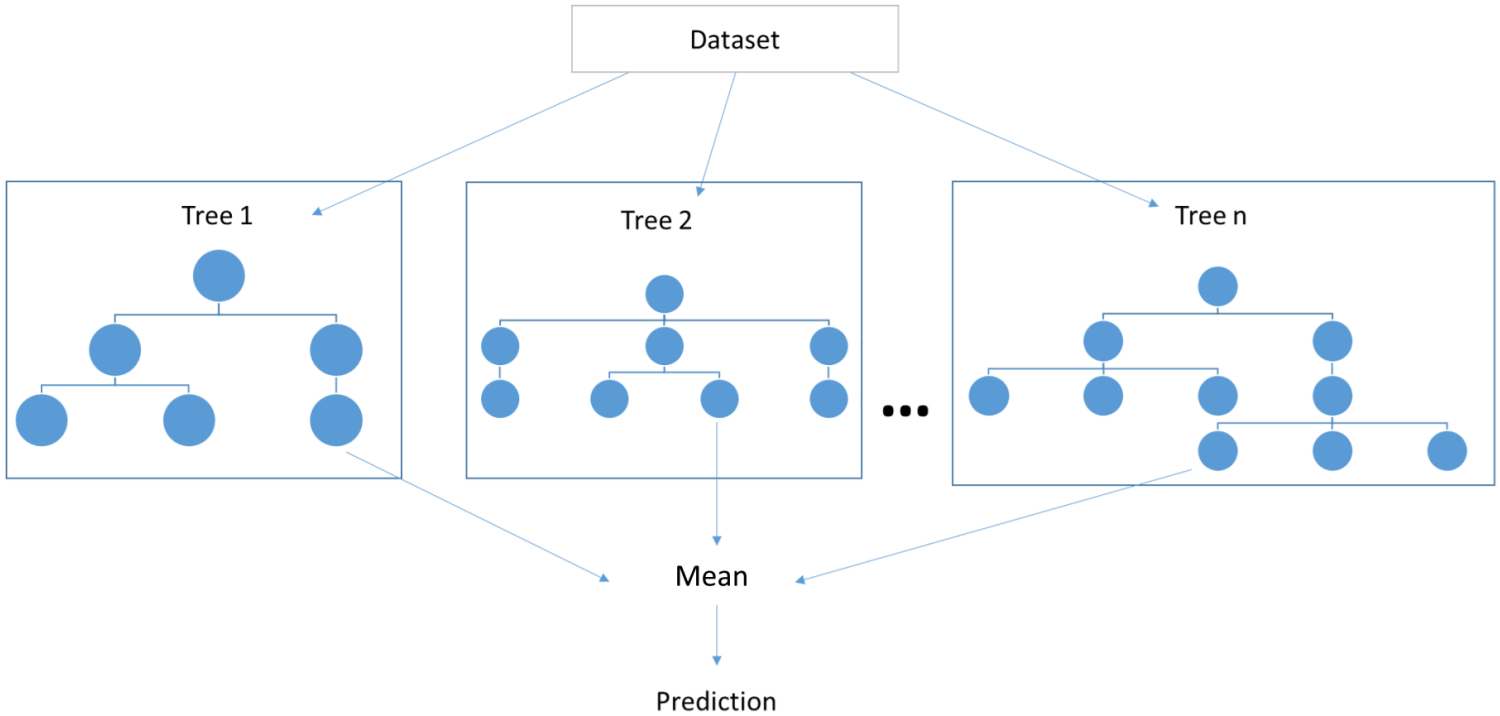


Figure: Random Forest Regressor

Random Forest

More about Random Forest:

- Random Forest differs from bagging decision trees because each of the trees are trained from a subset of the features with a process known as the Random Subspace method.
- Random Forest remedies a single decision tree's tendency to overfit the data and controls variance.
- Compared to Neural Networks, it can give good results with fewer data samples.
- Low computational cost

Random Forest

For our purposes, we used the Random Forest regressor from sklearn, which defaults to 100 trees and uses mean squared error as criterion to measure the quality of a split. (RMSE: 0.32)

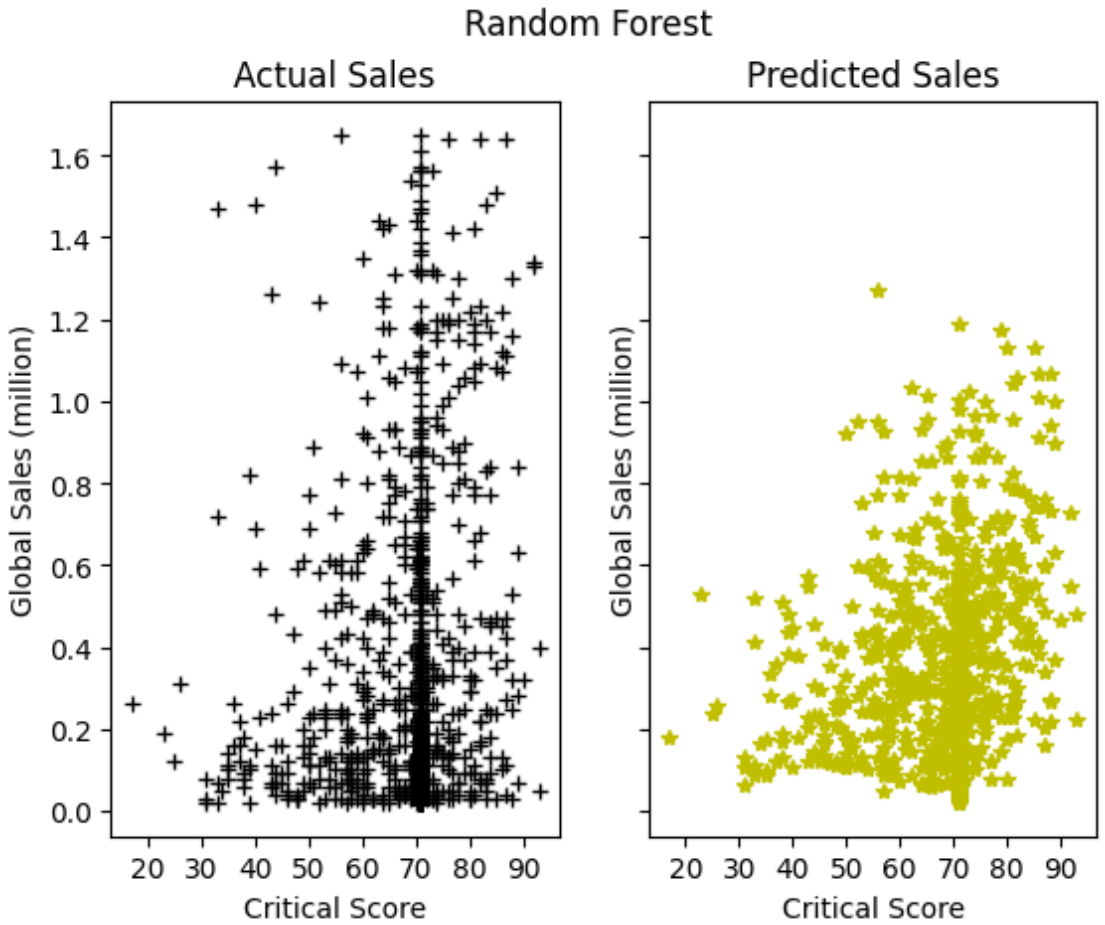


Table of Contents

Artificial Neural Network

Chen's slide.