# Sharif University of Technology
# Machine Learning in Physics project:
# Success in Movies
# Phase 2: Traditional Models

Ali Setareh Kokab , Reyhane Ghanbari

May 7, 2021

# Contents

**1 Summary** **3**

**2 Preparing the data** **3**

**3 models** **3**

3.1 k-nearest neighbors . . . . . 3
3.2 Polynomial regression . . . . . 5
3.3 SVR with polynomial kernel . . . . . 5
3.4 SVR with RBF kernel . . . . . 5
3.5 Random forest . . . . . 5

# 1 Summary

In the phase two of our project, we tried five classic models for our regression problem, predicting the success of movies. The models which we tried are KNN[1], polynomial regression and SVR[2] with two different kernels namely rbf and polynomial and the last model is random forest. For each model, we plot the learning curve and validation curve.

# 2 Preparing the data

Our goal is to predict the domestic gross of the movies **before** they release. As a result, we should only keep the features that are available before the date which a movie goes on screen. These features are: budget, opening theaters, runnig time (min), genre, age rating, day, month, language, country and the rank of the movie's cast. The system of ranking is explained in the previous article. Our prediction variable (Y) is domeestic gross. We take 2019 movies as our test data and take the rest as our training data set. We used PCA[3] to reduce the dimension of our data. We also use logarithm of domestic gross and budget instead of their values due to the high skewness of these two columns.

# 3 models

This part describes the regression models that we used on our data.

## 3.1 k-nearest neighbors

We used KNeighborsRegressor sklearn object for this part. The complexity of KNN models are determined by the number of neighbors that the regressor uses to estimate the best fit. To find the best number of neighbors we plot validation curve for this model fig 1. To find out number of neighbors in which the distance between validation curve and train curve are lowest, we plot the difference between these two (fig 2) and find the minimum. As it is seen, the best result obtains with k = 600.

After finding the best number of neighbors, we plot the learning curve fig 3. As is can be seen, about 2500 samples is enough to fit the model.The bias and variance of this model are equal to 3.92 and 0.1, repectivly.

[1] K-nearest neighbors
[2] support vector machine
[3] principal component analysis

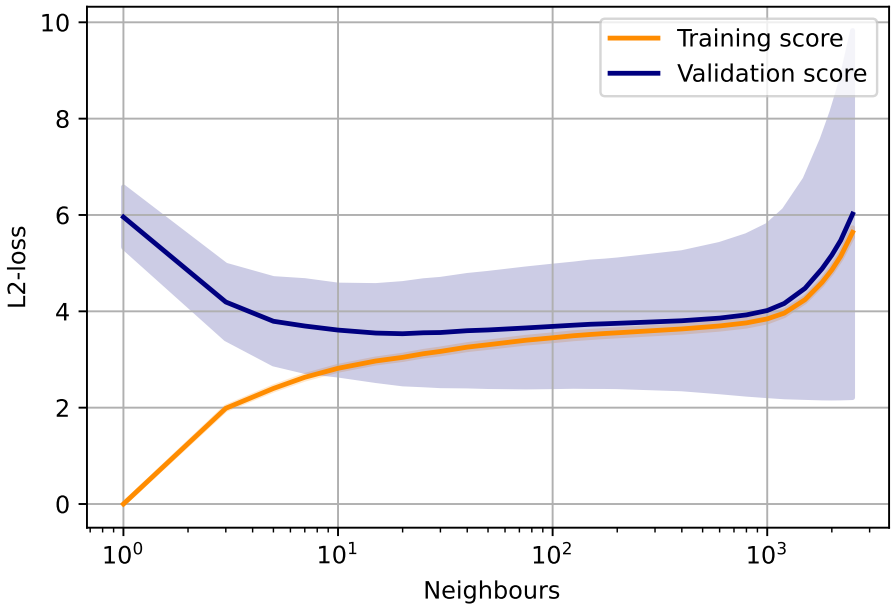


Figure 1: Validation curve for KNN model.

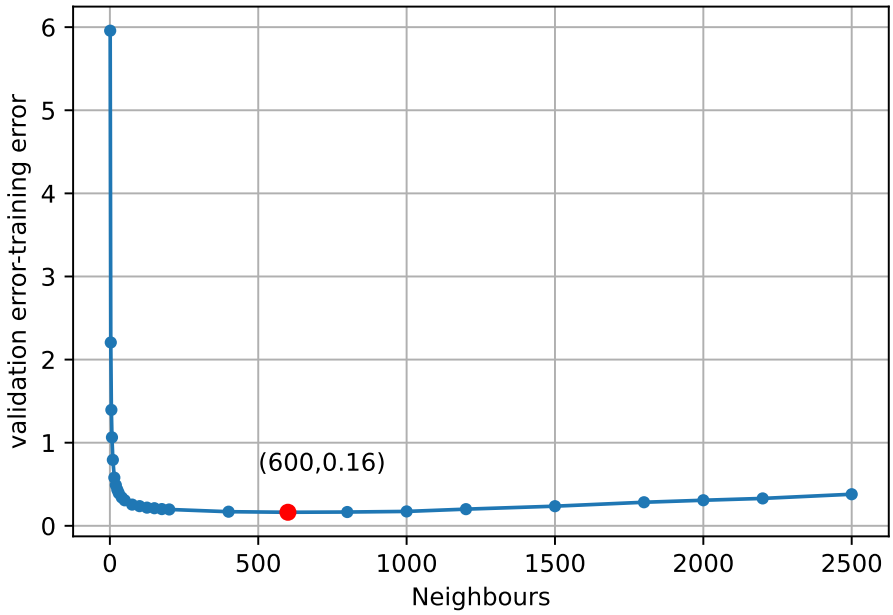


Figure 2: Difference between validation error and training error. The red point represents the optimum value.

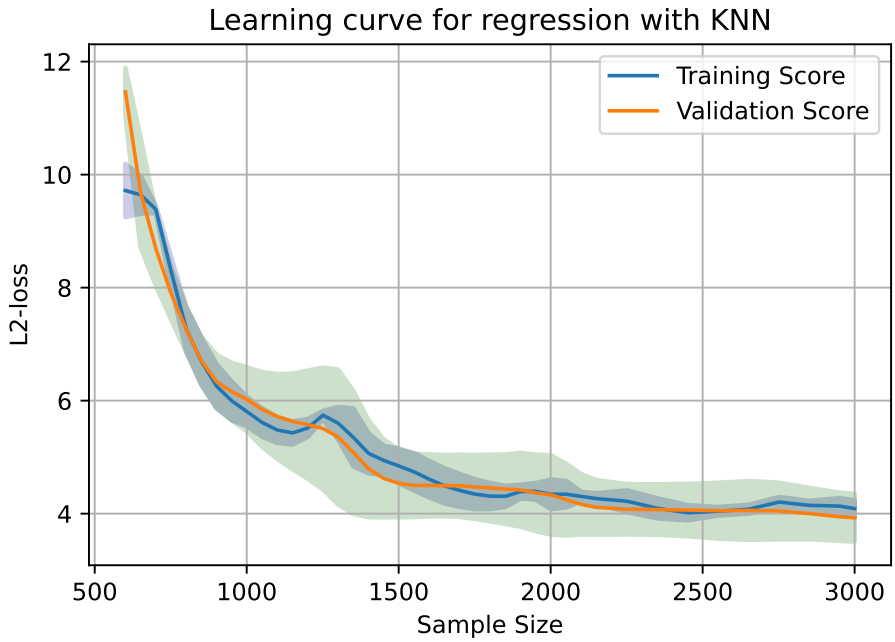


Figure 3: Learning curve for knn with k=600

### 3.2 Polynomial regression

### 3.3 SVR with polynomial kernel

### 3.4 SVR with RBF kernel

### 3.5 Random forest

We used RandomForestRegressor from sklearn object for this part. The complexity of Random forests are determined by the number of tress and maximum depth of the trees. First, we plot validation curve for a constant number of trees equal to 10 fig 4. Note that in this plot both validation and train loss decrease by increasing the depth of the tree. because of this, to pick the optimum value for the depth of the trees, we look at difference between these two curves and choose the least depth that this becomes constant which is the flat part of the fig 5. As it is seen, the optimum depth for this number of trees is about 30. At this point we get $R^2$ score 0.70 and mean squared error of 2.32 on the ttest data. In fig 6 you see the learning curve for this model.The bias and variance of this model are equal to 3.92.
As it is mentioned, random forests have two hyper parameters. In previous part we fixed the number of the trees and play with depth. Now we use sklearn grids search module to find the optimum depth and number of trees simultaneously.

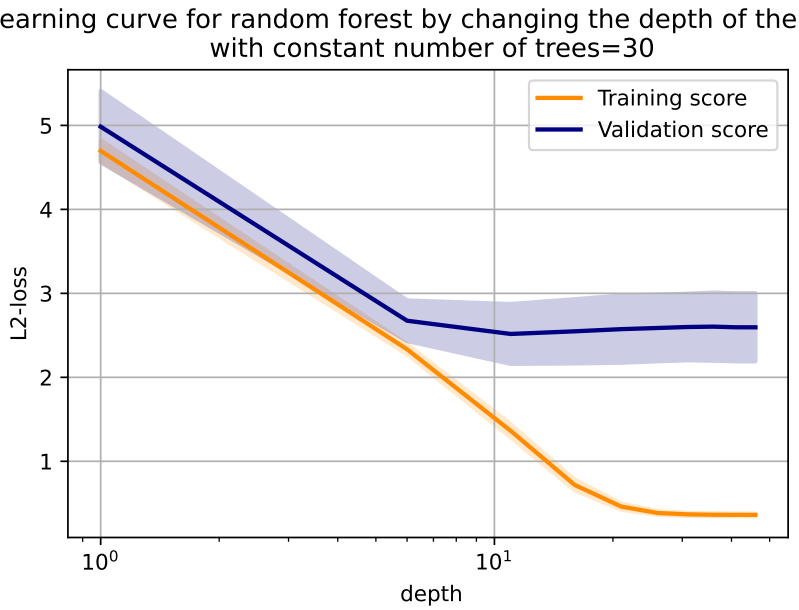


Figure 4: Validation curve for random forest model.

The optimum values for depth and number of trees are 9 and 70 respectively. For these values, mean squared error is 2.25 and $R^2$ score is 0.73. As you can see, our performance is slightly better than the previous part. You can see the learning curve for these values in fig 7.

# References

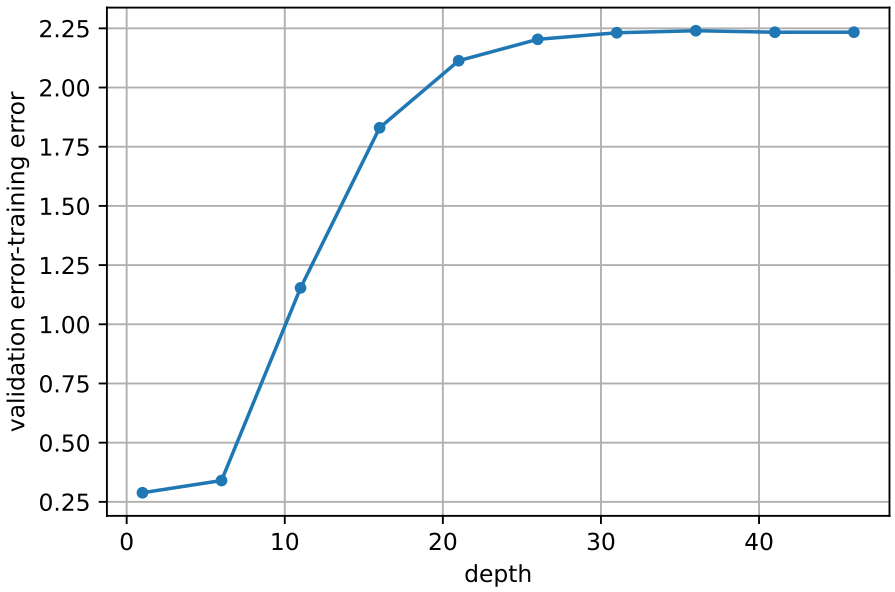


Figure 5: Difference between validation error and training error.

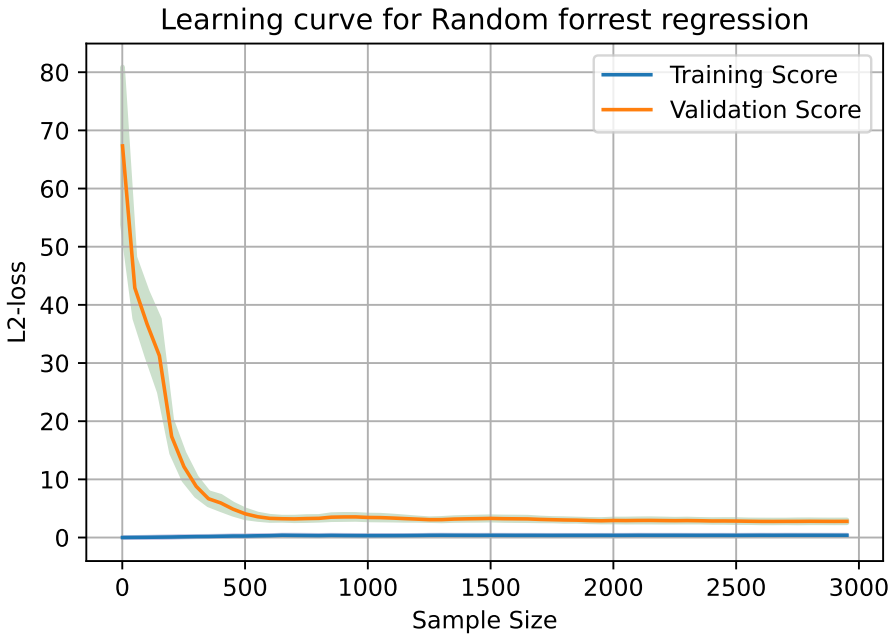


Figure 6: Learning curve for random forest 30 trees and maximum depth of 30

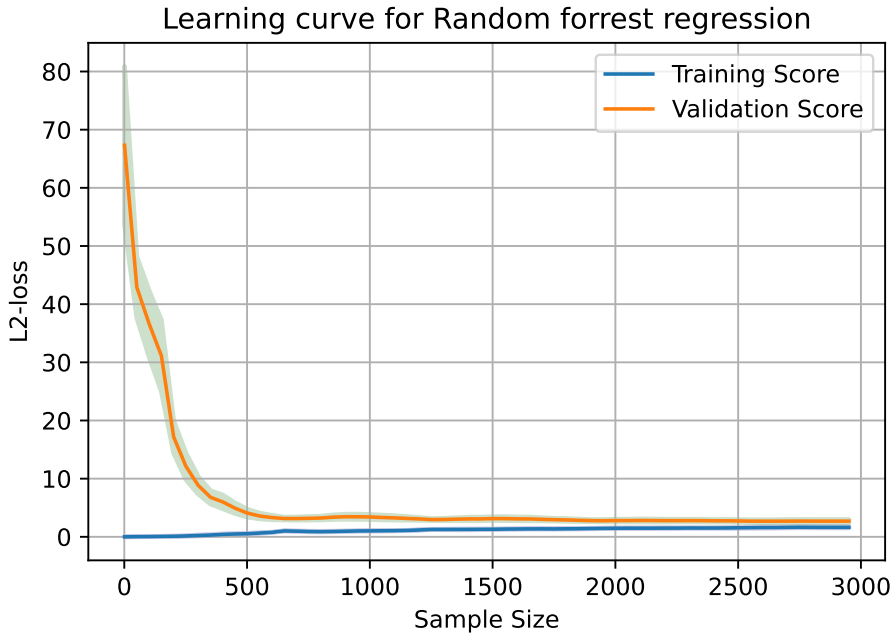


Figure 7: Learning curve for random forest with optimum values of 9 trees and maximum depth of 70. Note that in this figure the gap between validation and train is smaller in comparison with fig 6