Datamining and Machine Learning

Project: IMDB ratings

Ádám Gergely Szabó (QVJ2QB)

Fall Semester, 2021

Contents

1 Introduction

This project is about IMDB ratings, made available by [2], what models could be used for machine learning and how well these models predict ratings. In this project, regression models can be used as the outcomes are (almost) continuous. The work is done in Jupyter Notebook environment with python3 as its base. The different regression models are provided by sci-kit learn and the neural networks are made in tensorflow. Let's see how does the data set look.

2 Data Set Description

The data is provided by [2] which comes in a '.csv' format. It's easy to understand what is inside but in this form it cannot be utilized for machine learning.

Col.	Name	Date	Rate	Votes	Genre	Duration	Type
E.g.	No Time to Die	2021	7.6	"107,163"	Action Adventure Thriller	163	Film

Col.	Certificate	Ep.	Nudity	Violence	Profanity	Alcohol	Frightening
E.g.	PG-13	-	Mild	Moderate	Mild	Mild	Moderate

Table 1: The different columns in the given data. All of them are strings when loading, conversion is needed.

2.1 How to Handle Each Column

Generally, most columns are easy to convert to some kind of numeric type. The 'Name' column obviously unneeded as it is specific to given film, can be thrown away. The 'Date' column states the year of release. The 'Nudity','Violence','Profanity','Alcohol' and 'Frightening' columns are still easy to understand but need some encoding and they can be ordered from 'None' to 'Moderate'. The 'Type' columns actually has two types: 'movie' or 'series' and according to this, the 'Episodes' column as a number or '-' as movies don't have episodes.

The Certificate column is a different kind. The rated movie or series are coming from 1922 to 2022 so their ratings are coming from different sysems of certificates. For example, in the early days they only approved films. Now, for different age and target

groups we have PG/GP (which stands for genera audiences) and E (exempt). These sometimes overlap, so should be grouped together. Also, these can be ordered and for the models used, the encoding should be ordered too.

Original	New
Approved, E, TV-G, G	G
TV-PG, PG, GP	GP
M/PG, M	M
TV-MA, NC-17, X, R	R
TV-Y7-FV, TV-Y7	TV-Y7
(Banned)	(Banned)
PG-13	PG-13
TV-14	TV-14

Table 2: The different columns in the given data. All of the are strings when loading, conversion is needed.

The hardest one to handle is the Genre column. A rated movie and series can have one to three different Genres. Even though the data contains $\sim$ 6000 rows, there are only 377 unique genres present. My idea was that these should be expanded into 'isAction' type columns, because there are only 27 different type of genre.

After this, the data is ready for usage. The target columns is the Rate column which contains the different ratings from 0.0 to 10.0. Of course, there no such thing as a perfect 10 present in the data. There are rows that have missing values and generally these have values that has "No rate" in place and as such, easy to remove. It was a heavier task to sort out the data and order it.

3 Applied Machine Learning Methods

This data contains review ratings that rounded to the first decimals, so I will use regression models that do predictions that ends with values coming from a continuous interval. Yet again, these values are rounded to first decimals and they are from 0.0 to 10.0 which means 100 possible ratings. My idea was to do and try to predict to original ratings and as such, I will round to predicted values to the first decimals and compare that to the original ratings. Of course, the option for classification exists as we could classify the different ratings into groups (e.g. "good" or "bad"). The problem is that even though the data consists of 6000 rows, not all ratings are present from the [0.0,10.0] interval, thus encoding the ratings themselves for classification is not a solution.

In the following, I will fit models and compare the predicted rating to the original rating. The accuracy of each model are calculated by comparing the rounded prediction to the original, the R^2 score is calculated with the predicted values and the original ratings.

3.1 K Nearest Neighbour Regression

My initial guess was that the ratings can predicted by looking and the closest K number of neighbours of the given point. This model came with a somewhat high accuracy and R^2 score. After this, I used the standard scaled data to see how much this affect the prediction accuracy. From my experience, standard scaling decreases accuracy for KNNR and that's what happened too. (Figures: 1 & 2)

3.2 Linear and Polynomial Regression

After KNNR, I wanted to go through many regression models and the next one was linear regression. Polynomial regression was achieved by using a pipeline for linear regression. My expectation was that these model won't perform as well as KNNR and the results prove me right: these models may be too simple to grasp the many variables that could be needed for correct prediction. (Figures: 3 & 4)

3.3 Ridge and ElasticNet

Both of these models come with different hyperparameters that could be used to tune these models forward a model that is better than linear regression. Even after

tuning these, the results were not satisfying. For these models, I used the scaled data. (Figures: 5 & 6)

3.4 Decision Tree Regression

This model comes from the family of ensemble models. This means that it doesn't matter if the data has ordered encoding (e.g. columns 'Nudity' where the different levels of nudity can be ordered) and doesn't matter if the scaled or not. This model creates trees that can be led to fully grown trees. In the following, I created 2 different DTR models according to their maximum depth: 11,25. The lower value led to higher R^2 score, while the higher depth resulted in higher prediction accuracy. (Figures: 7 & 8)

3.5 Simple Neural Network with Tensorflow

I was quite interested in using neural network for predicting ratings as they seem to be superior to other models in their accuracy. But some research had to be done in order to build a neural network that does regression and some tuning were needed for achieving high accuracy. For the models, I used MeanSquaredLogarithmicError() and SquaredHinge() from keras as others didn't perform well. With these two, the performances are similar. The network is the following: (Figures: 9)

```
model: Sequential()
> Dense, 1024, kernel_initializer = 'normal', selu activation
> Dropout, 0.07
> BatchNormalization

> Dense, 512, kernel_initializer = 'normal', selu activation
> Dropout, 0.07
> BatchNormalization

> Dense, 512, kernel_initializer = 'normal', selu activation
> Dropout, 0.07
> BatchNormalization

> Dense, 1, kernel_initializer = 'normal', linear activation
```

1: The used model for prediction

4 Conclusion

It was interesting to see the different models performance compared to each other. Doing classification with the given data may have given a higher accuracy, because the ratings are not that continuous. It was somewhat frustrating that I was unable to improve the neural network further to show its superiority over other models.

The results always have below 0.5 R^2 score which means that something hidden is happening. As it turns out in [1] it is hard to correctly predict human behaviour and higher than 50% may be unachievable.

References

[1] Jim Frost. How high does r-squared need to be? https://statisticsbyjim.com/regression/how-high-r-squared/.

[2] Mazen Ramadan. Imdb most popular films and series. https://www.kaggle.com/mazenramadan/imdb-most-popular-films-and-series.

5 Appendix

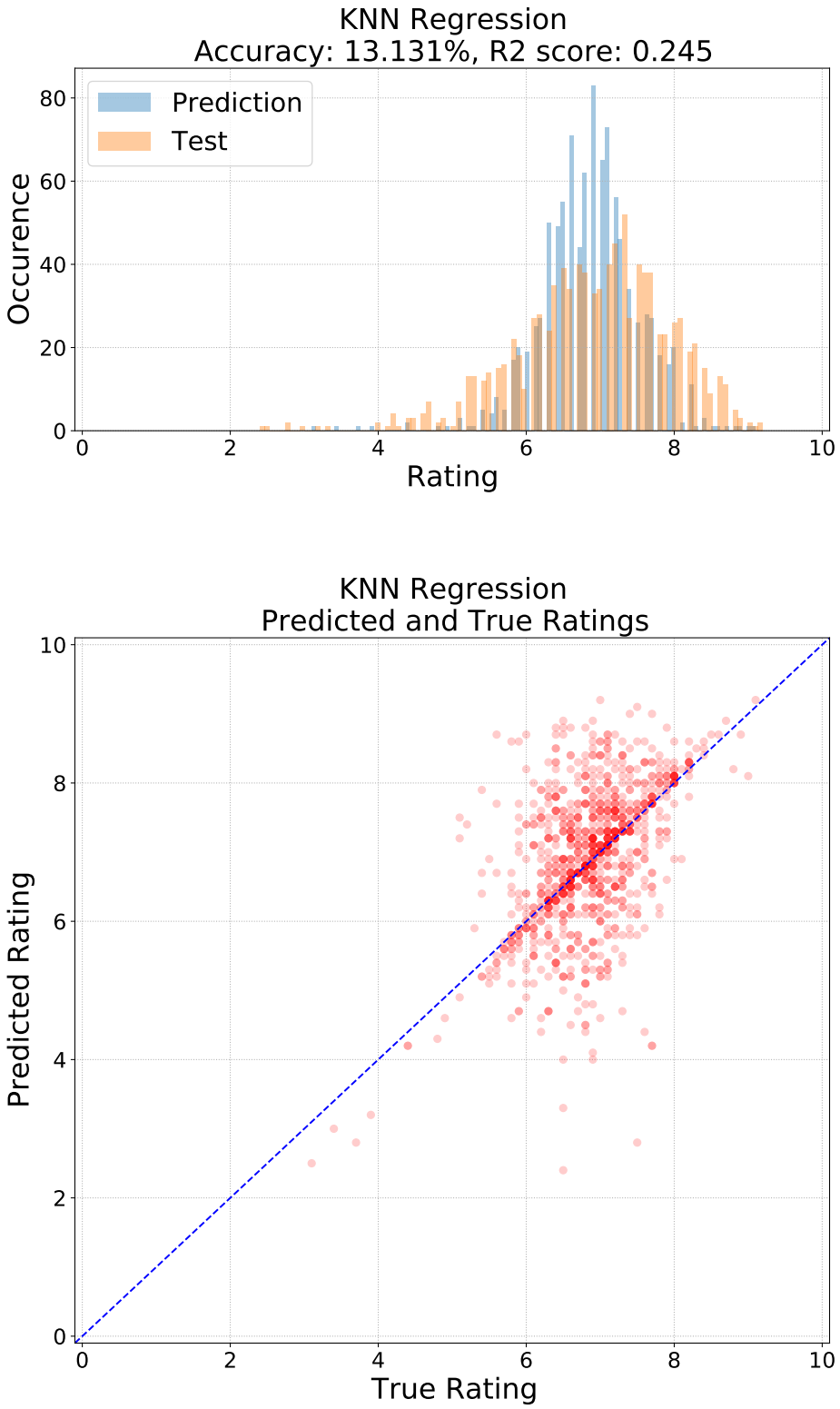


Figure 1: KNN Regression. Section at 3.1

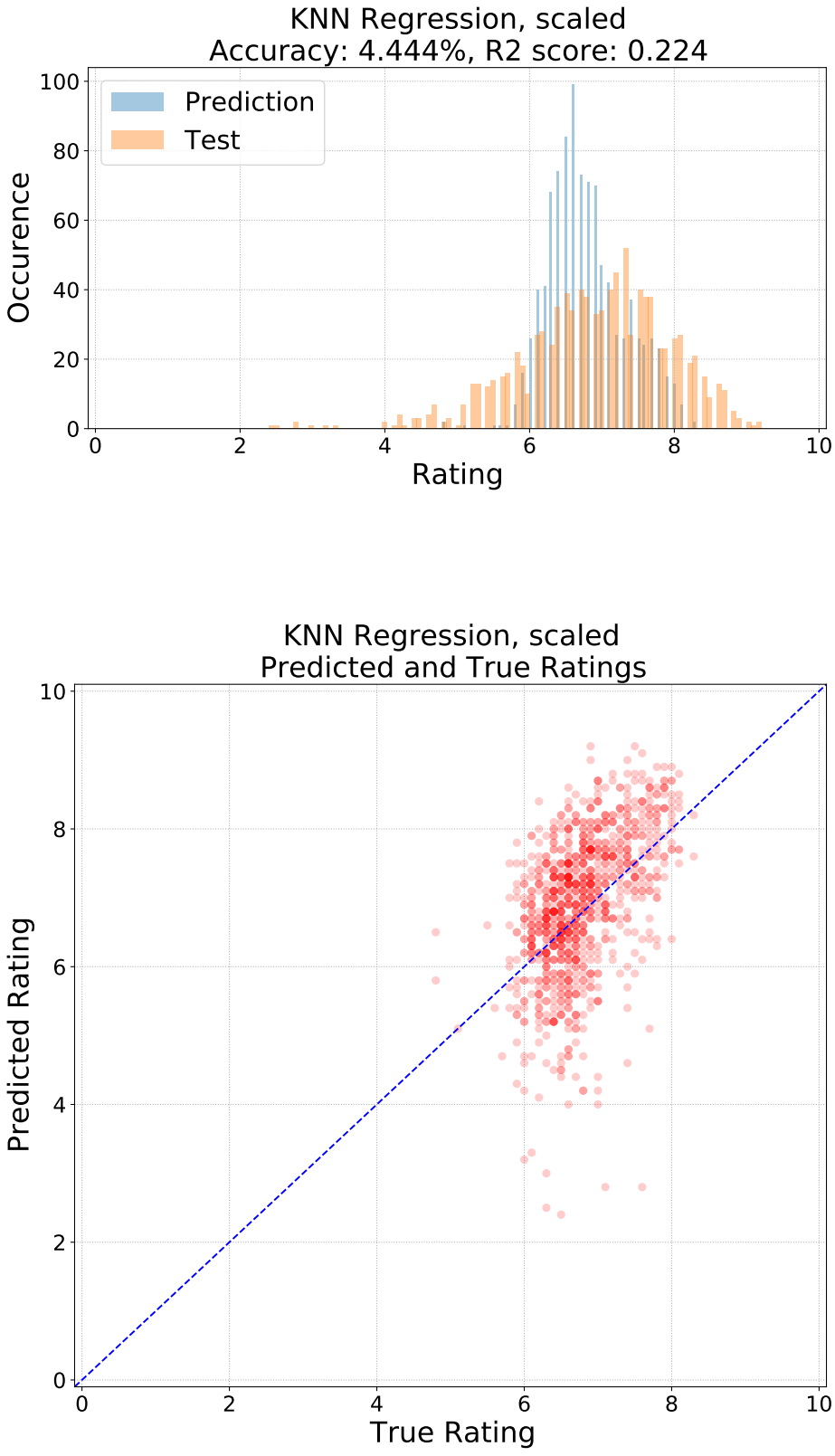
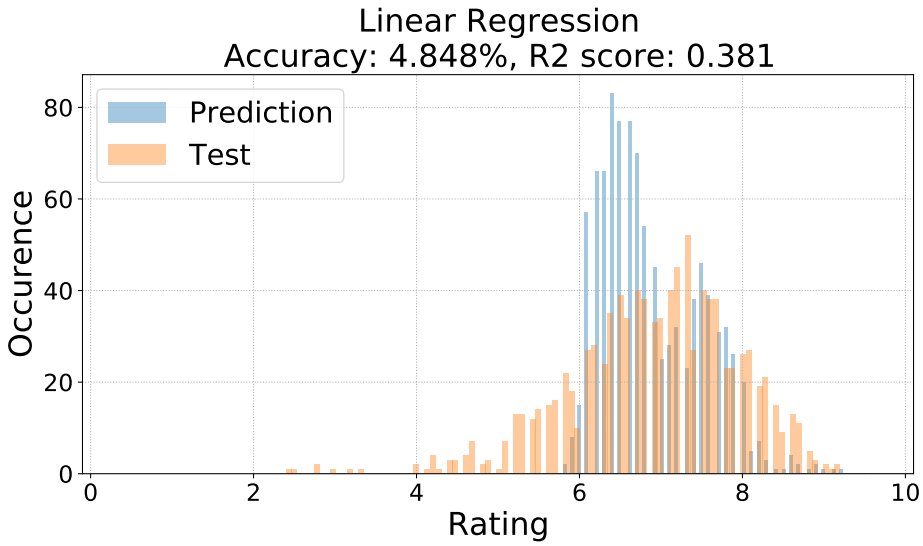


Figure 2: KNN Regression, scaled. Section at 3.1



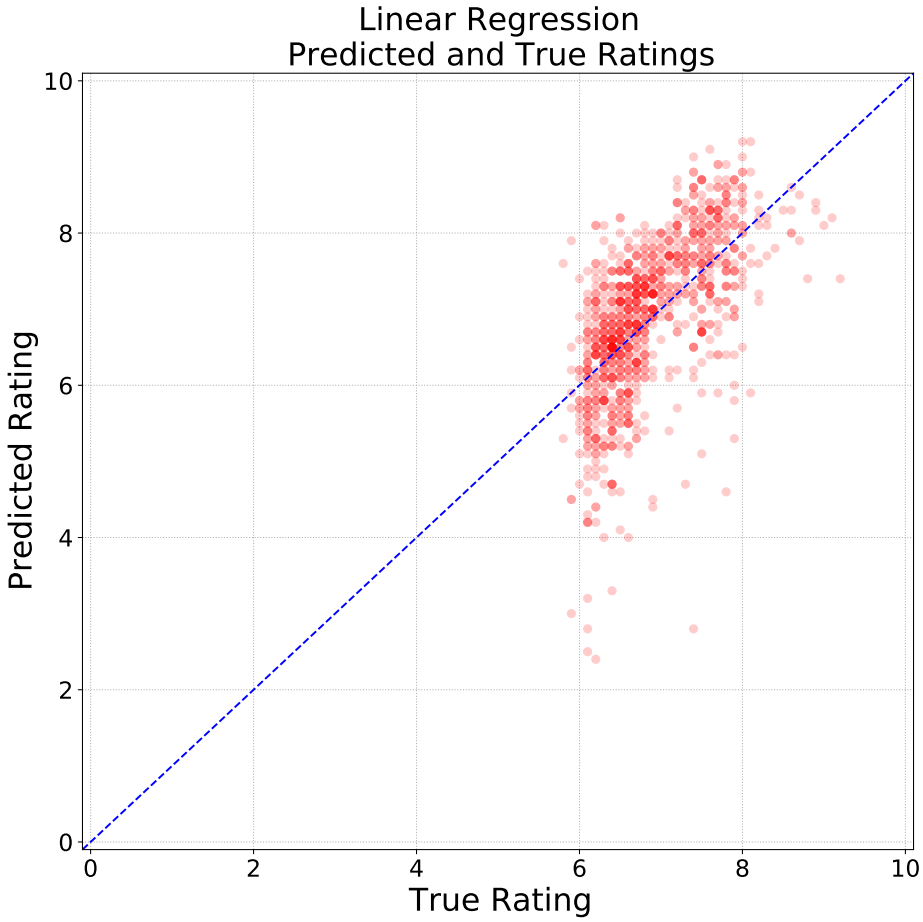


Figure 3: Linear Regression. Section at 3.2

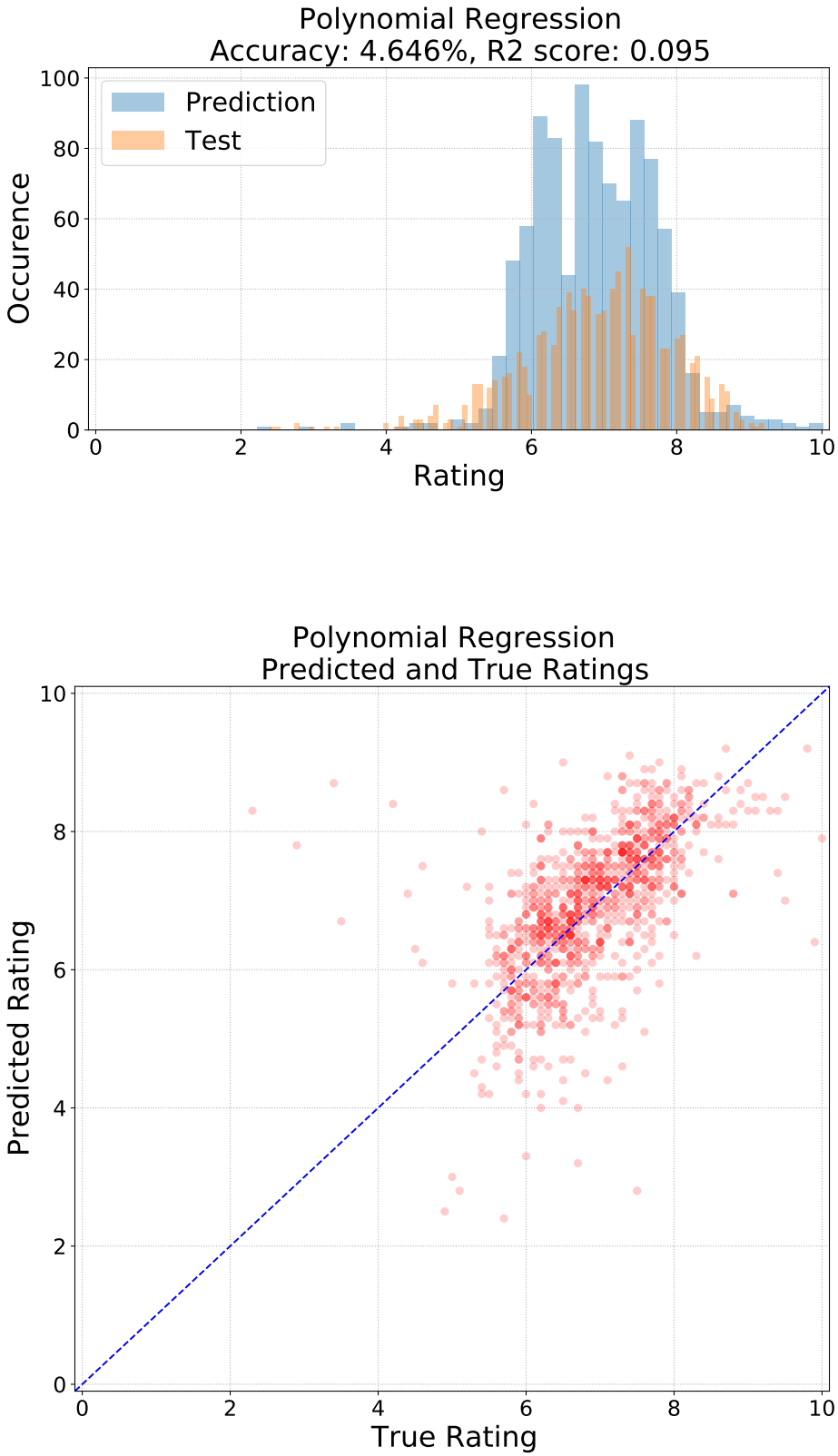
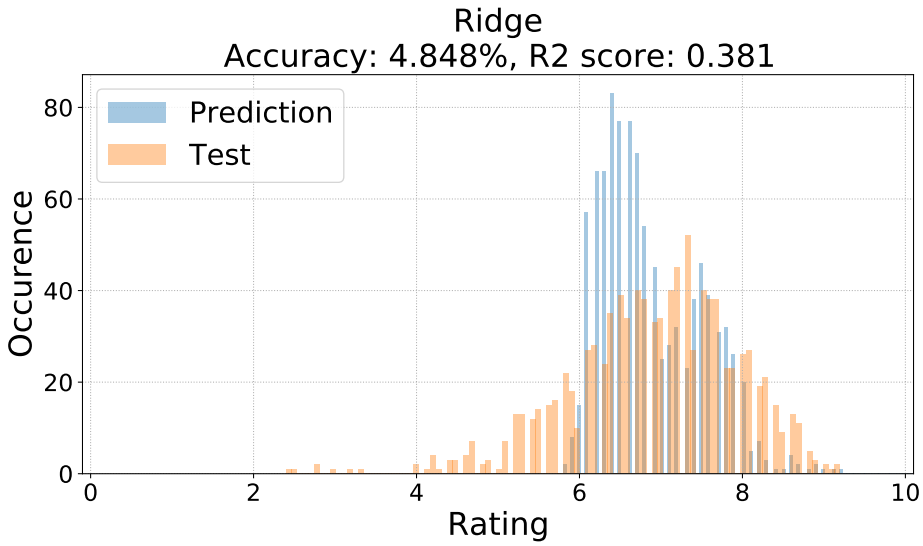


Figure 4: Polynomial Regression. Section at 3.2



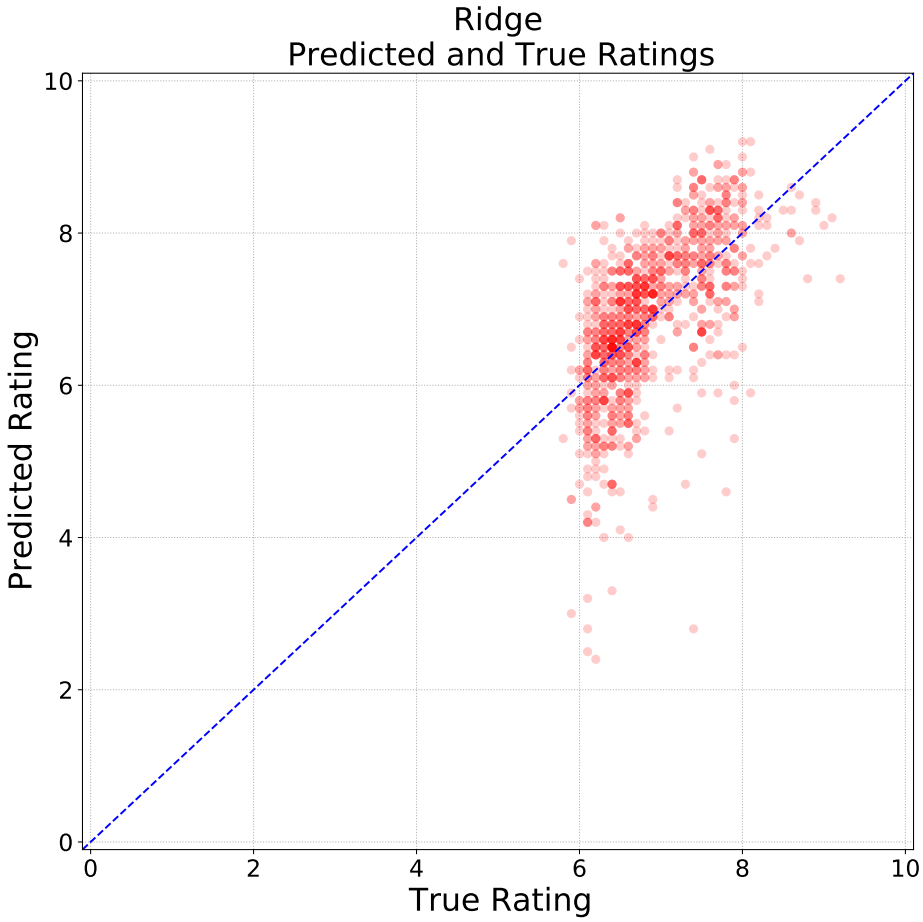
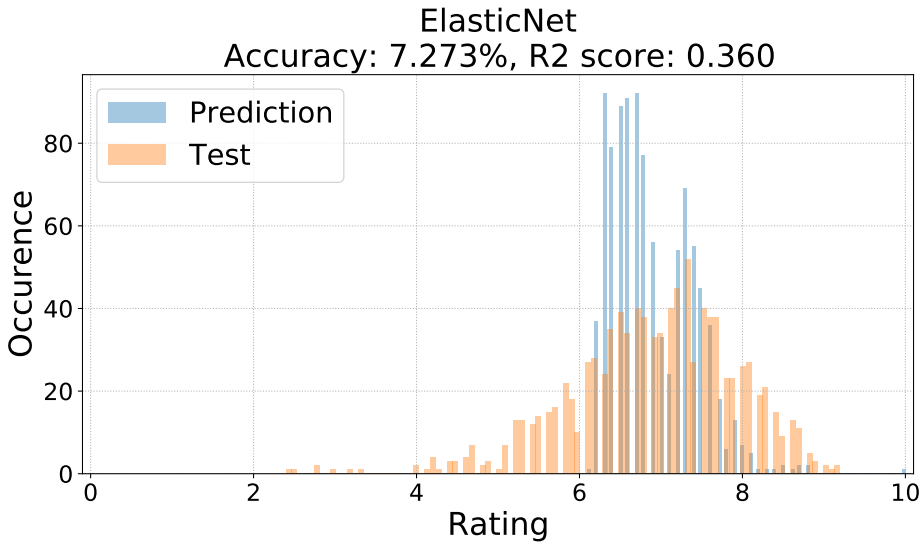


Figure 5: Ridge. Section at 3.3



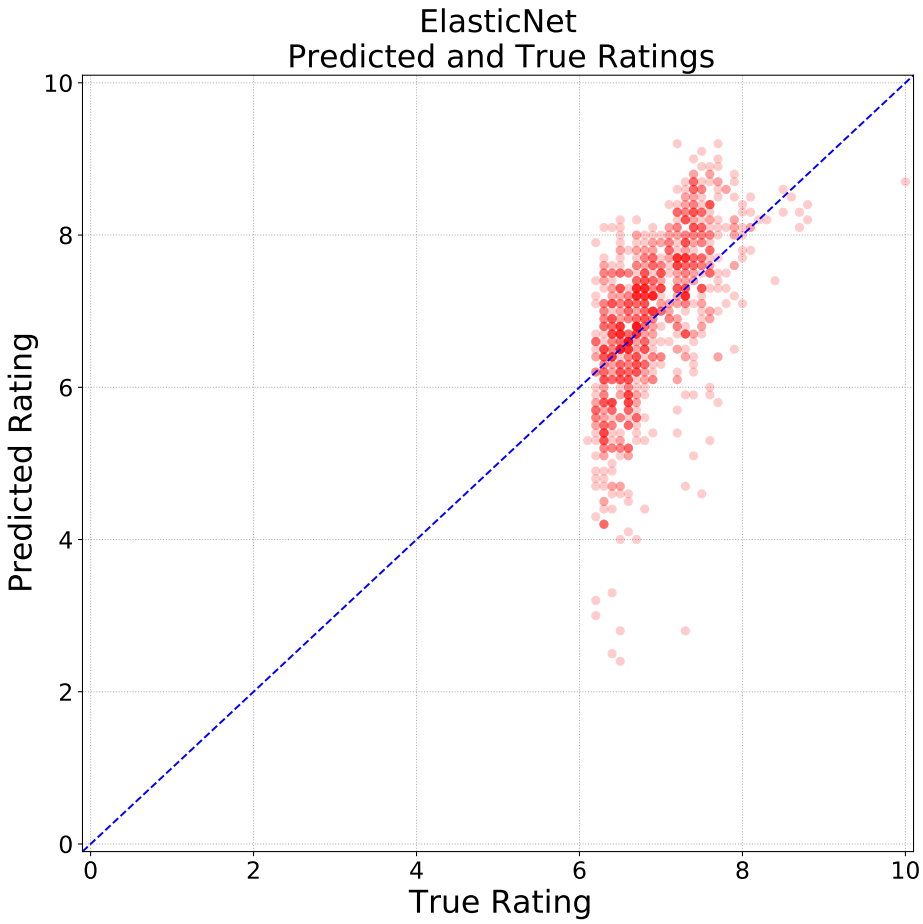


Figure 6: ElasticNet. Section at 3.3

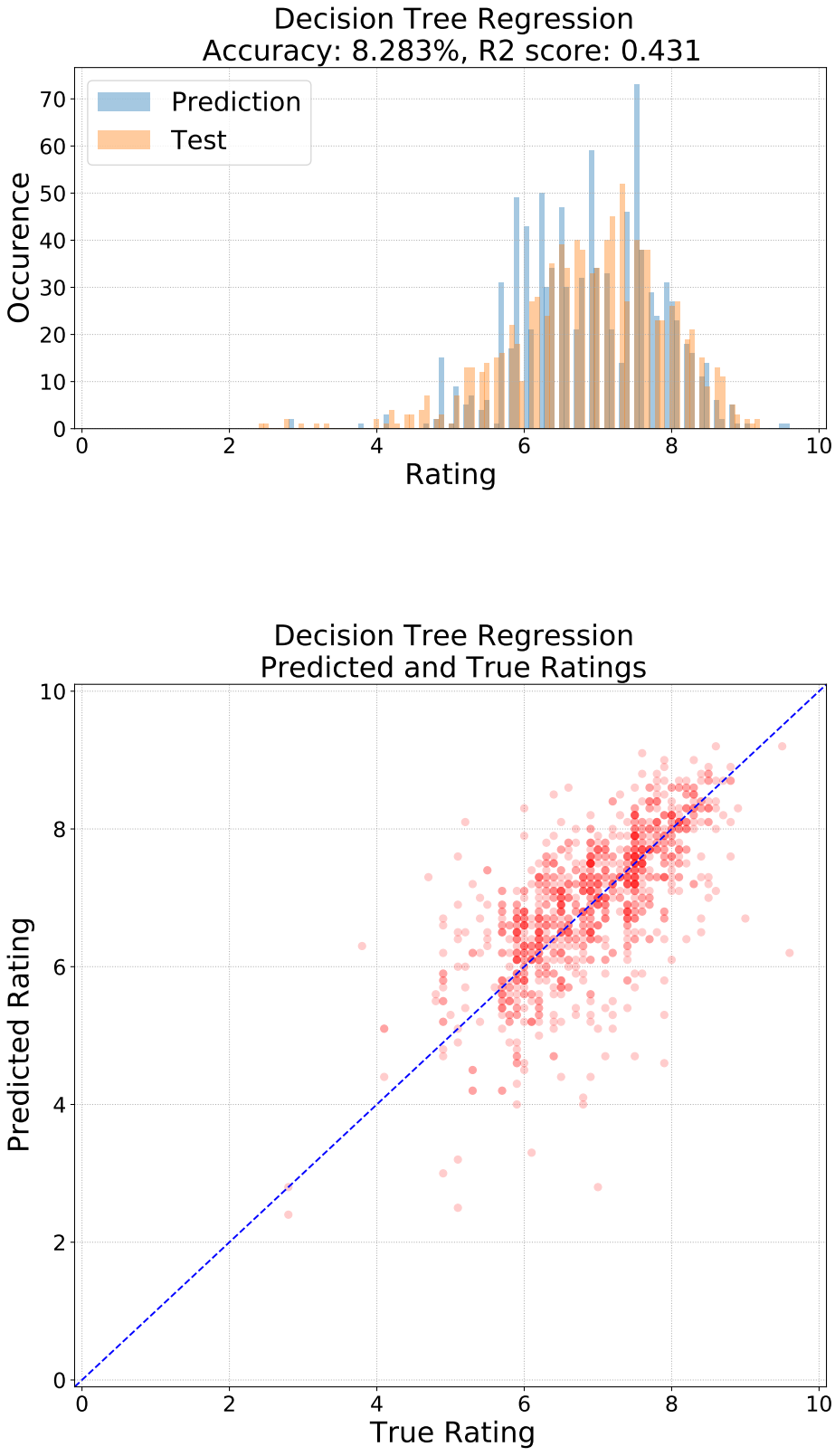


Figure 7: Decision Tree Regression with 11 as max depth. Section at 3.4

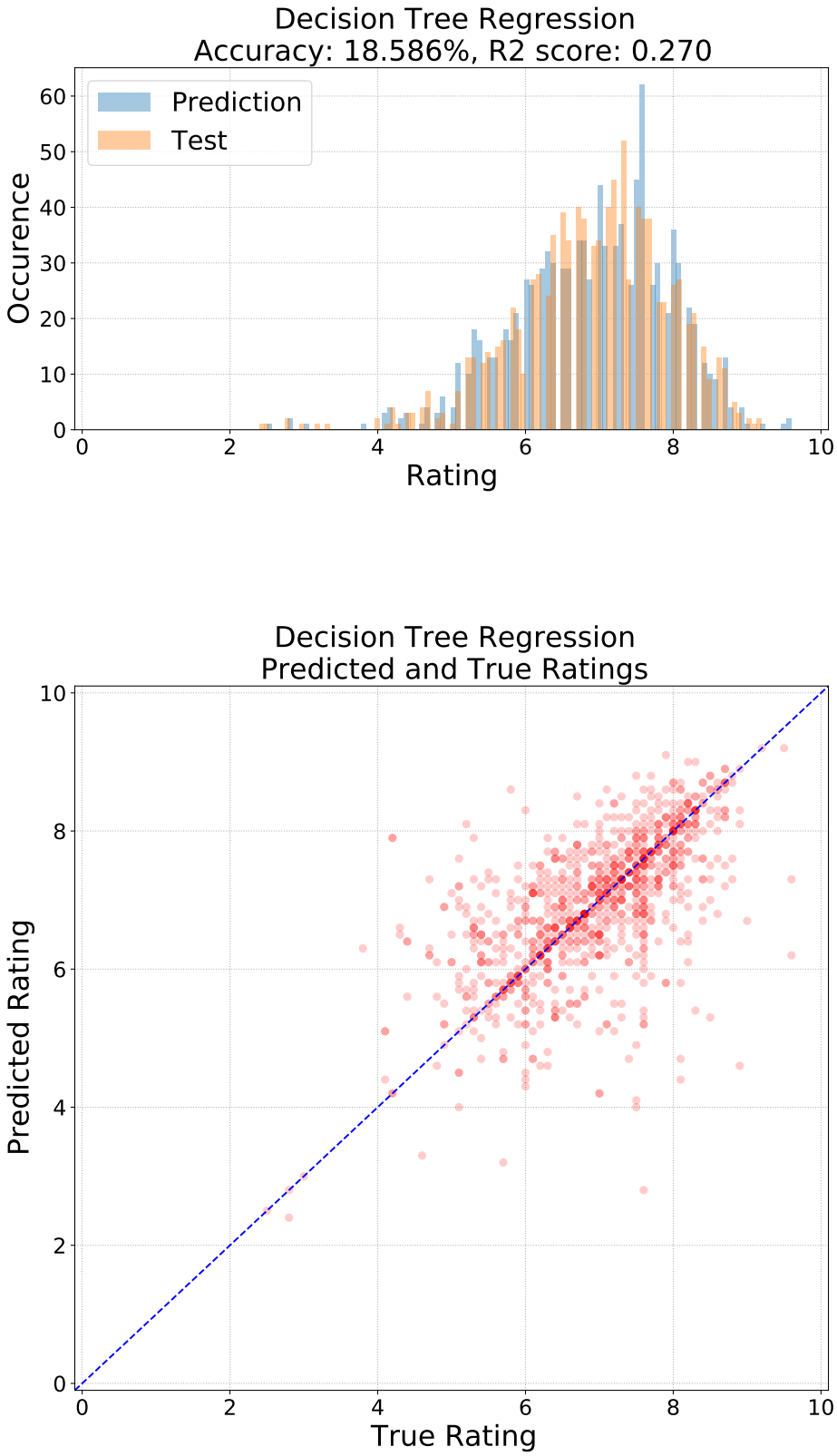
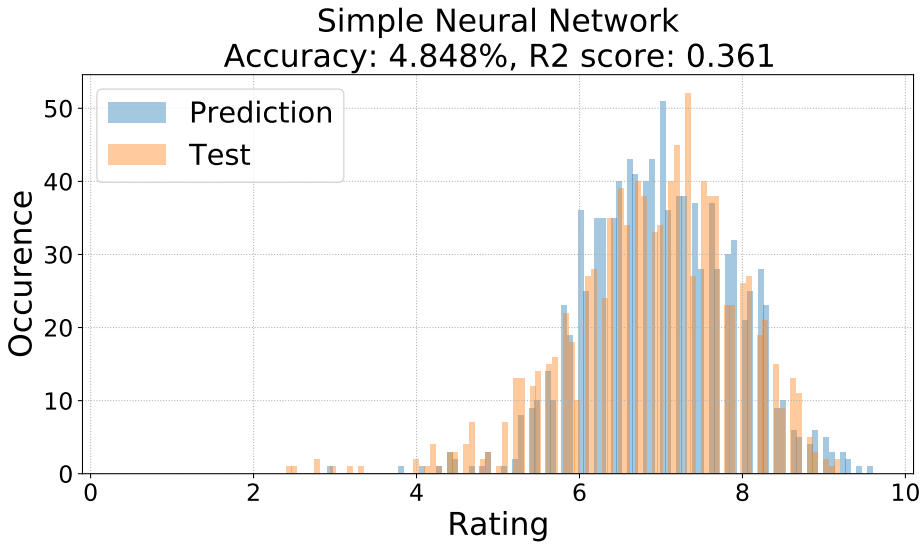


Figure 8: Decision Tree Regression with 25 as max depth. Section at 3.4



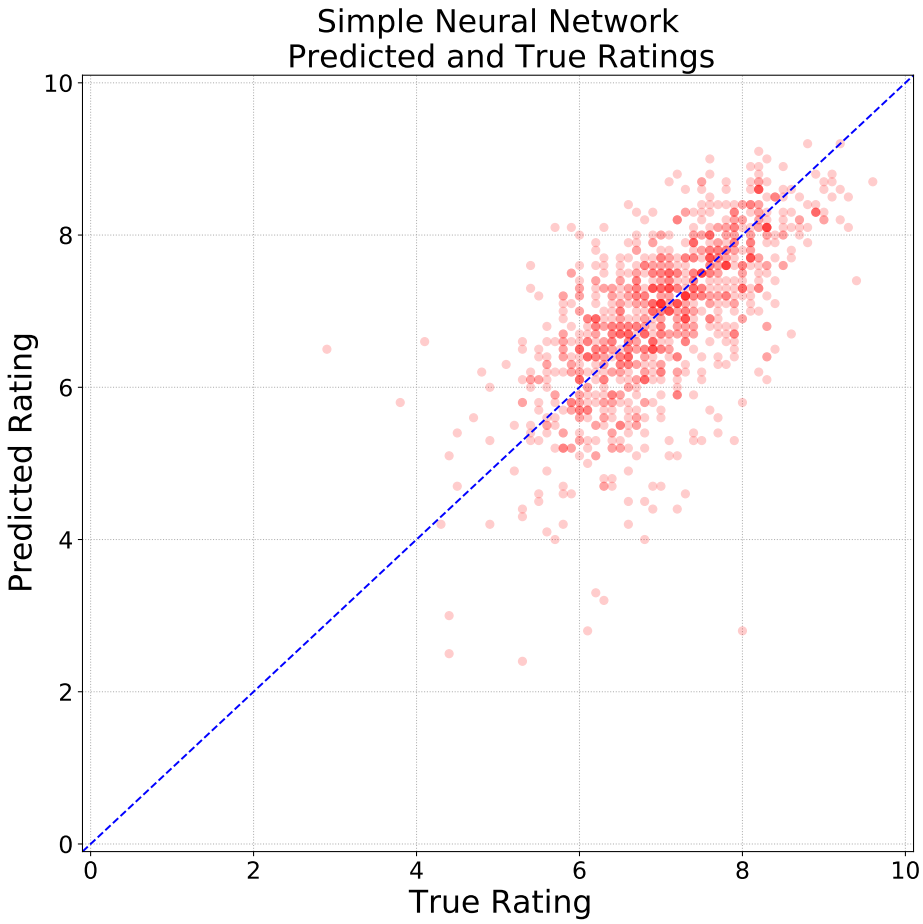


Figure 9: Simple Neural Network. Section at 3.5