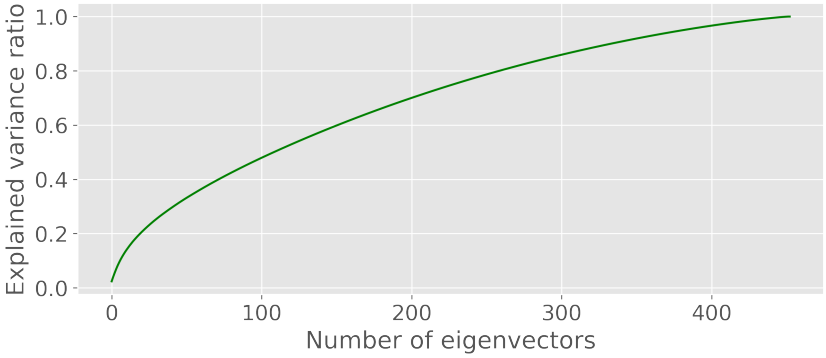
Training the model

We would like to predict the prices of games using a regression model. We will in the following scale our features to zero mean and unit variance. Afterwards we will perform a principal component analysis (PCA) in order to reduce the amount of features taken into consideration and thus we will account for a degree of overfitting. We will like to keep 90% of the variance in our training data. This is done by looking at the explained variance ratio of each feature and keeping the features with most explained variance ratio until we reach our threshold of 0.90. Afterwards we will fit an elastic net-model to our training data using 10-fold cross validation and perform a grid search to tune our hyperparameters of the elastic net-model. The hyperparameters are λ and α. λ is the penalty term that penalizes the amount of features in our model and α determines the convex combination of LASSO penalization and Ridge penalization.
First we take a look at our dataset. Our target is 'price'. We drop the features 'name', 'release_date', 'developer' and 'id' to create our feature matrix. Then we split our data into training and test date with a 80-20 ratio. Then we scale our training data. This is done to make sure that features of different magnitude are taken equally account for when fitting our model. As we have 453 features to begin with one can easily imagine that a lot of our features would be made redundant without scaling, as we have many binary features and several numeric features of different magnitude in our data. The scaling is done with the sklearn.preprocessing.StandardScaler class. Next we perform a principal component analysis on our training data. We start by including 453 components. This way we can for each component see how much of the explained variance each component contributes to the full variance of the training data. Below we see a figure of the cummulative sum of explained variance ratio as a function of the number of eigenvectors to include



To keep our threshold of 0.90% of the variance, we find that we must have 334 components in our final model. We see that we lower our dimension of features by approximately one fourth. PCA is done to avoid too much overfitting while still maximizing variance in the dataset. The PCA is done with the sklearn.decomposition.PCA class. In the following we will use features and components interchangably but we stress that after performing a PCA we have reduced our feature matrix $\boldsymbol{X}$ of dimensionality $N \times 453$ into a component matrix $\boldsymbol{Z} = \boldsymbol{XU}$ where $\boldsymbol{U}$ has dimensions 453×334 thus $\boldsymbol{Z}$ has dimension $N \times 334$.
We will now look at LASSO regression and Ridge regression seperately. Regression is a supervised learning method that seeks to find weights for all features such as to minimize the predicted target of an observation and the true value. A prediction

has the following form

$$\hat{y} = \boldsymbol{w}^\intercal \boldsymbol{x}$$

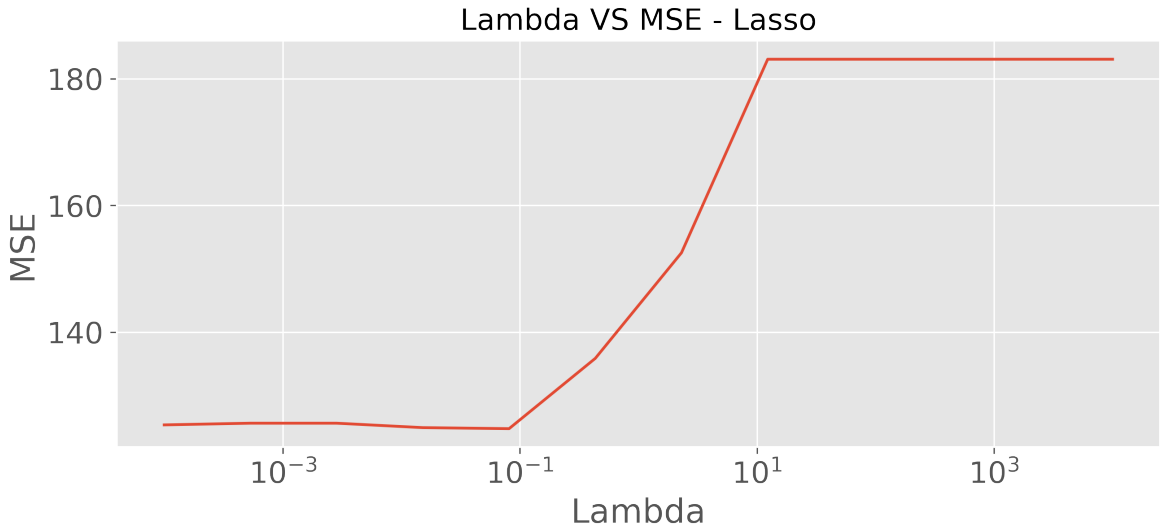
(INCLUDE REFERENCE TO PML, CHAP 10, PDF PAGE 451)Both LASSO and Ridge penalize the amount of features we keep in a final model. LASSO regression seeks to minimize the following function:

$$J(w)_{LASSO} = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} |w_m|$$

Note that N is the number of observations, m is the number of features and $\lambda > 0$ is a hyperparamter. The first sum is referred to as the sum of squared errors (SSE). The second is our penalty term. LASSO penalizes the absolute value of our weights. To tune our hyperparameter λ we perform 10-fold cross validation on our training data with different values of λ, where

$$\lambda \in [0.0001, 10000]$$

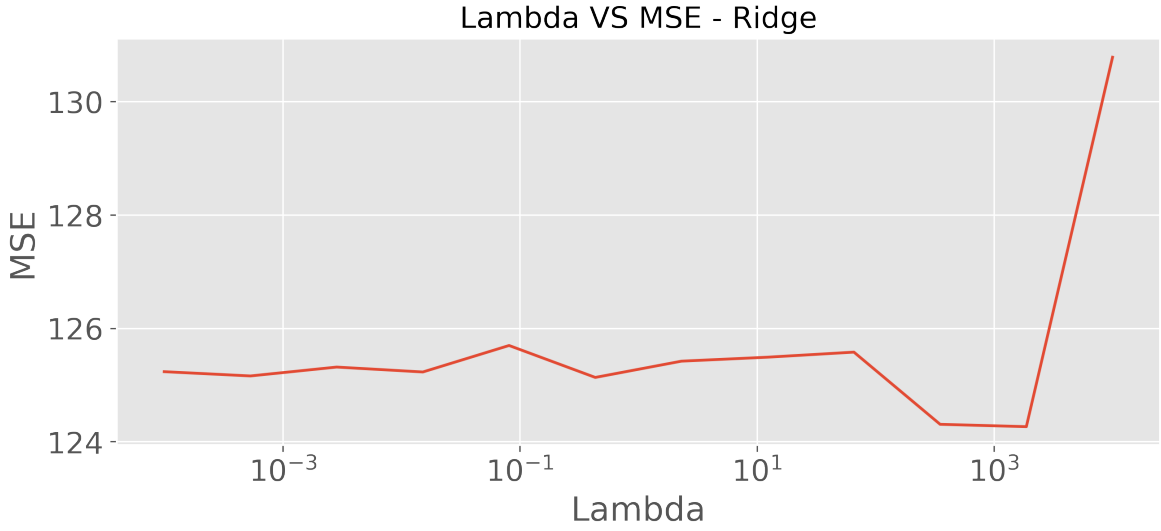
For each value of λ we compute the mean squared error (MSE) on the validation set and averaging over all 10 folds. Performance for each λ can be seen here



Note the optimal λ, $\lambda^*_{LASSO} = 0.081113$. Training a new model with λ^*_{LASSO} on all of our training data and predicting on our training data yields an MSE of 116.3. Ridge regression seeks to minimize another function quite similar to the LASSO, namely:

$$J(w)_{Ridge} = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} w_m^2$$

The SSE is the same but Ridge penalizes the squared value of our weights. We do the same procedure as previously described to tune our hyperparameter, λ. We search within the same space as before and performance for each λ can be seen here



The optimal λ is found at $\lambda^*_{Ridge} = 1873.8$. Training a new model with λ^*_{Ridge} on all of our training data and predicting on our training data yields an MSE of 115.8. All in all we see that the two types of regressions yield approximately the same MSE but two very different orders of magnitude for the optimal hyperparameter. In order to combine the two different types of regression we introduce the Elastic Net model. Elastic Net regression seeks to minimize a convex combination of the two penalty terms and looks like the following:

$$J(w)_{ElasticNet} = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m}(\alpha|w_m| + (1-\alpha)w_m^2)$$

In order to tune a model with two hyper parameters, λ and α, we perform a grid search. Since the optimal λ for the two previous models were of different magnitudes we still search in the same area but we decrease the step size in order to search through more values of λ. Since α is a convexity coefficient, we search between values of α where

$$\alpha \in [0, 1]$$

We also include 10-fold cross validation within the grid search. To perfrom the grid search we use the sklearn.model_selection.GridSearchCV class. We find that the optimal λ, $\lambda^*_{ElasticNet} = 0.0001$ and optimal α, $\alpha^*_{ElasticNet} = 0.05263$. Predicting on our training data yields an MSE of 115.4. Overall the performance of our Elastic Net model is only slightly better which easily can be random due to the way the training data is split during the cross validation. Ontop of that the computational time increases dramatically as our grid search seeks to optimize in two dimensions. To sum up we have the following results and hyper parameters

	λ	α	MSE
Lasso	0.08113	-	116.3
Ridge	1878.8	-	115.8
Elastic Net	0.0001	0.05263	115.4

One could argue that a possible better model should include polynomial features to cater non-linear relationships in the dataset. As our data includes 334 components, a polynomial expansion of second order would consist of 56.279 components. Trying to fit a Ridge regression model without cross validation with fixed hyperparameter

takes 1 minute and 53 seconds. Considering an Elastic Net model with 10-fold cross validation and a grid of 400 gridpoints would at least take roughly 4.000 times longer than fitting the Ridge regression model which is why we have left out the possibility of polynomial expansion in our final model.