CMPT726 - Machine Learning
Assignment 2

Topic: Probabilistic Reasoning, Maximum Likelihood, Classification
Submitted to: Professor Oliver Schulte
Submitted by: Raquel Yuri da Silveira Aoki
Date: October 24, 2017

Question 1 - Variance and Covariance

For definitions and notation please refer to the text. We write var(X) for the variance of a single random variable and $cov(X, Y)$ for the covariance of two random variables, such that $var(X) = cov(X, X)$.
1) Show that $var(X) = E(X^2) - [E(X)]^2$.

Solution
Covariance definition: $cov(X, Y) = E[(X - E(X))(Y - E(Y))]$
Then:

$$\begin{aligned}
var(X) &= cov(X, X) \\
&= E[(X - E(X))(X - E(X))] \\
&= E[XX - 2XE[X] + E(X)^2] \\
&= E[XX] - 2E[X]E[X] + E(X)^2 \\
&= E(X^2) - 2E(X)^2 + E(X)^2 \\
var(X) &= E(X^2) - [E(X)]^2 \qquad \text{(Eq. 1)}
\end{aligned}$$

2) Show that if two random variables X and Y are independent, then their covariance is zero.

Solution

$$\begin{aligned}
cov(X, Y) &= E[(X - E(X))(Y - E(Y))] \\
&= E[XY - XE(Y) - YE(X) + E(X)E(Y)] \\
&= E[XY] - E(X)E(Y) - E(Y)E(X) + E(X)E(Y) \\
&= E(X)E(Y) - 2E(X)E(Y) + E(X)E(Y) \text{ Using f(x,y)=f(x)f(y)} \\
cov(X, Y) &= 0
\end{aligned}$$

(Eq. 2)

It was possible to use f(x,y)=f(x)f(y) because X and Y are independent.

Question 2 - Decision Tree Learning

1) Install a package that implements the ID3 decision tree algorithm that we studied in class, for both discrete and continuous input features. We recommend using Weka, see course web page.

Solution: It was used Weka to solve this question.

2) Apply the ID3 learner to the hockey draft dataset, using GP > 0 as the target class variable. For data preprocessing, drop the sum_7yr_GP column.

Solution The data was preprocessing excluding the column 'sum_7yr_GP'. It was used as training set the years 2004, 2005 and 2006, and for the testing set the year 2007.

In order to avoid overfitting, it was used as stop condition the size of the node. If the leaf has less than 11 elements, then the node not split anymore. The value 11 was chosen after some tests, where this value presented the highest accuracy.

3) Show the decision tree learned. Which branch is the most informative, meaning that its leaf has the lowest class entropy? Given your understanding of the domain, do the features on the branch make sense?

The leaf with the lowest class entropy is 'rs_G > 15: yes'(36,0), with entropy equal to 0 ($-\frac{36}{36}log(1) - \frac{0}{36}log(0) = 0$). To find out the split with the lowest entropy, we just have to search between the leafs because $E(parents) \geq E(child_1) + .. + E(child_n)$.

The Table 1 shows the entropy of the six leaf nodes. Using this table it is possible to notes that CSS_rank is the feature that produces a split with the lowest entropy.

Table 1: Leaf nodes entropy

Split	Nodes	Entropy
po_A ≤ 0	(11,2)+(15,6)	0.8495
rs_PlusMinos ≤ 0	(23,4)+(13,4)	0.7474
po_GP ≤ 6	(16,3)+(14,5)	0.8101
rs_G ≤ 14	(15,6)+(12,3)	0.8999
Weight ≤ 206	(47,14)+(20,7)	0.8952
CSS_rank ≤ 61	(14,6)+(19,1)	0.5892

The features used in the splits make sense. For example, CSS_rank represents the 'Central scouting service ranking in the draft year', and it is very likely that players in the first positions played a game in player's 7 years of NHL career. The next two splits are rs_G(Goals in regular seasons in the draft year) and rs_P(Points in regular seasons in the draft year), and it is also expected that players who made many goals or points during the regular season will also have 'GP_7yr_greater_than_0=yes'. In general, all features that represents Games Played, Goals, Points, Assists in regular or playoffs are good to fit 'GP_7yr_greater_than_0=yes' and they all are showed at the Decision Tree.

```
=== Run information ===

Scheme:       weka.classifiers.misc.InputMappedClassifier -I -trim -W weka.
    classifiers.trees.J48 -- -C 0.25 -M 11 -A
Relation:     a2_train_weka
Instances:    637
Attributes:   19
              DraftAge
              country_group
              Height
              Weight
              Position
              DraftYear
              CSS_rank
              rs_GP
              rs_G
              rs_A
              rs_P
              rs_PIM
              rs_PlusMinus
              po_GP
              po_G
              po_A
              po_P
              po_PIM
              GP_greater_than_0
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

InputMappedClassifier:

J48 pruned tree
------------------

CSS_rank <= 12
|   rs_G <= 15
|   |   rs_PlusMinus <= 0: yes (23.0/4.0)
|   |   rs_PlusMinus > 0: no (13.0/4.0)
|   rs_G > 15: yes (36.0)
CSS_rank > 12
|   rs_P <= 11: no (99.0/16.0)
|   rs_P > 11
|   |   rs_PlusMinus <= 0
|   |   |   rs_PlusMinus <= -1
```

```
|   |   |   |   country_group = EURO: no (32.0/2.0)
|   |   |   |   country_group = CAN
|   |   |   |   |   DraftYear <= 2005
|   |   |   |   |   |   rs_G <= 15
|   |   |   |   |   |   |   po_A <= 0: no (11.0/2.0)
|   |   |   |   |   |   |   po_A > 0: yes (15.0/6.0)
|   |   |   |   |   |   rs_G > 15: yes (15.0/3.0)
|   |   |   |   |   DraftYear > 2005: no (15.0/4.0)
|   |   |   |   country_group = USA: yes (11.0/3.0)
|   |   |   rs_PlusMinus > -1
|   |   |   |   CSS_rank <= 39: yes (34.0/3.0)
|   |   |   |   CSS_rank > 39
|   |   |   |   |   rs_GP <= 32: no (22.0/5.0)
|   |   |   |   |   rs_GP > 32
|   |   |   |   |   |   country_group = EURO: yes (30.0/7.0)
|   |   |   |   |   |   country_group = CAN
|   |   |   |   |   |   |   DraftAge <= 18: no (14.0/5.0)
|   |   |   |   |   |   |   DraftAge > 18
|   |   |   |   |   |   |   |   po_GP <= 6: yes (16.0/3.0)
|   |   |   |   |   |   |   |   po_GP > 6: no (14.0/5.0)
|   |   |   |   |   |   country_group = USA
|   |   |   |   |   |   |   Height <= 71: yes (18.0/5.0)
|   |   |   |   |   |   |   Height > 71
|   |   |   |   |   |   |   |   rs_G <= 15: yes (15.0/6.0)
|   |   |   |   |   |   |   |   rs_G > 15: no (12.0/3.0)
|   |   rs_PlusMinus > 0
|   |   |   po_A <= 10
|   |   |   |   po_PIM <= 23
|   |   |   |   |   country_group = EURO: no (51.0/4.0)
|   |   |   |   |   country_group = CAN
|   |   |   |   |   |   rs_GP <= 57: no (12.0)
|   |   |   |   |   |   rs_GP > 57
|   |   |   |   |   |   |   Weight <= 206: no (47.0/14.0)
|   |   |   |   |   |   |   Weight > 206: yes (20.0/7.0)
|   |   |   |   |   country_group = USA
|   |   |   |   |   |   CSS_rank <= 61: yes (14.0/6.0)
|   |   |   |   |   |   CSS_rank > 61: no (19.0/1.0)
|   |   |   |   po_PIM > 23: yes (16.0/5.0)
|   |   |   po_A > 10: yes (13.0/1.0)

Number of Leaves  :   27

Size of the tree :  50

Attribute mappings:

Model attributes                          Incoming attributes
------------------------------------      --------------------
(numeric) DraftAge                 --> 1 (numeric) DraftAge
(nominal) country_group            --> 2 (nominal) country_group
(numeric) Height                   --> 3 (numeric) Height
(numeric) Weight                   --> 4 (numeric) Weight
(nominal) Position                 --> 5 (nominal) Position
(numeric) DraftYear                --> 6 (numeric) DraftYear
(numeric) CSS_rank                 --> 7 (numeric) CSS_rank
(numeric) rs_GP                    --> 8 (numeric) rs_GP
(numeric) rs_G                     --> 9 (numeric) rs_G
(numeric) rs_A                     --> 10 (numeric) rs_A
```

```
(numeric) rs_P                    —> 11 (numeric) rs_P
(numeric) rs_PIM                  —> 12 (numeric) rs_PIM
(numeric) rs_PlusMinus            —> 13 (numeric) rs_PlusMinus
(numeric) po_GP                   —> 14 (numeric) po_GP
(numeric) po_G                    —> 15 (numeric) po_G
(numeric) po_A                    —> 16 (numeric) po_A
(numeric) po_P                    —> 17 (numeric) po_P
(numeric) po_PIM                  —> 18 (numeric) po_PIM
(nominal) GP_greater_than_0       —> 19 (nominal) GP_greater_than_0


Time taken to build model: 0.02 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances          134               70.1571 %
Incorrectly Classified Instances         57               29.8429 %
Kappa statistic                           0.3957
Mean absolute error                       0.3813
Root mean squared error                   0.4634
Relative absolute error                  76.7673 %
Root relative squared error              92.6911 %
Total Number of Instances               191

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  Class
                 0,792    0,400    0,690      0,792    0,737      no
                 0,600    0,208    0,720      0,600    0,655      yes
Weighted Avg.    0,702    0,309    0,704      0,702    0,702

=== Confusion Matrix ===

  a  b   <-- classified as
 80 21 |  a = no
 36 54 |  b = yes
```
Listing 1: WEKA Output

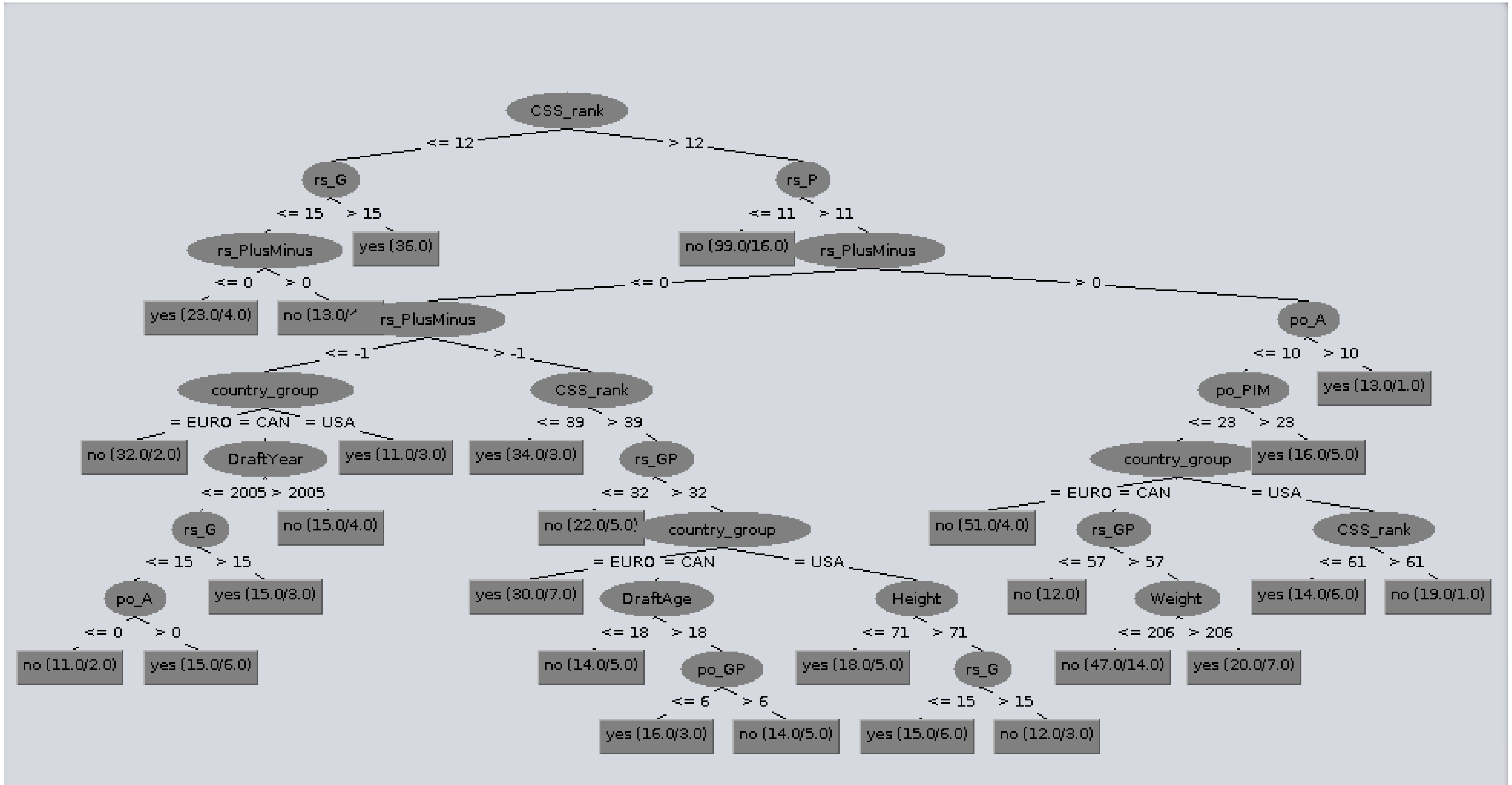

Figure 1: Decision Tree learned at weka

4) (Bonus Question) Rerun the Naive Bayes classifier from assignment 1 on the new training and test set for this assignment. Compare the test set accuracy of the decision tree learner to the result of the Naive Bayes classifier.

Solution Using the Naive Bayes classifier developed in Assignment 1, with the years 2004, 2005 and 2006 on the training set and 2007 in the testing set, we obtain the follow accuracy:

Table 2: Naive Bayes - 2004/2005/2006 Training and 2007 Testing

Observed	Predicted	
	Yes	No
Yes	65 (34.03%)	25(13.09%)
No	52 (27.23%)	49 (25.65%)

Comparing the Decision Tree Weka output showed at Listing 1 and the Naive Bayes accuracy at Table 2, it is possible concluded that, for the period between 2004 and 2007, Decision Tree is a better predictor, once that its accuracy is 70.15%, versus 59.68% in the Naive Bayes.

Question 3 - Minimum Least Squares Error for Regularized Linear Regression

Consider least-squares linear regression with L2 regularization as defined in the text.
1) Using the notation of the text, write down the squared-error function, including the regularization term.

Solution

$$E_X^\lambda(w) = \frac{1}{2}\sum_{n=1}^{N}(y_n - x_n \bullet w)^2 + \frac{\lambda}{2}||w||^2$$

$$E_X^\lambda(w) = \frac{1}{2}(Y - X \bullet w)^T(Y - X \bullet w) + \frac{\lambda}{2}w^T w \qquad \text{(Eq. 3)}$$

2) Show that the weight vector w^* that minimizes this error function is given by $w^* = (\lambda I + X^T X)^{-1} X^T y$

Using the propriety: $\frac{\partial(Y-X\bullet w)^T(Y-X\bullet w)}{\partial w} = -2X^T(Y - X \bullet w)$

$$\frac{\partial E_X^\lambda(w)}{\partial w} = \frac{1}{2}(-2X^T(Y - X \bullet w)) + \frac{2\lambda}{2}w$$

$$\text{Using max(f(x)) = min(-f(x))}$$

$$X^T(Y - X \bullet w) = \lambda w$$

$$X^TY - X^TXw = \lambda w$$

$$X^TXw + \lambda w = X^TY$$

$$(\lambda I + X^TX)w = X^TY$$

$$(\lambda I + X^TX)^{-1}(\lambda I + X^TX)w = (\lambda I + X^TX)^{-1}X^TY$$

$$\text{Using} A^{-1}A = I$$

$$w^* = (\lambda I + X^TX)^{-1}X^TY \qquad \text{(Eq. 4)}$$

Question 4 - Practice: Implement Least Squares Regression

We will gain practice with linear regression by applying it to predict the number of NHL games that a player will have played after 7 years. So the independent target variable will be *sum_7yr_GP*. We will go through a few typical steps for a regression analysis: data preprocessing, weight learning, and model evaluation. We can increase the power of linear regression by adding non-linear terms as new derived columns. The functions that give rise to the new columns are called basis functions, and the new data matrix that includes the basis functions is called the design matrix. A common type of non-linear term to add are products of the original features that combine information from different columns; these are called interaction terms.

1) Data Preprocessing. In addition to what is specified on the website, apply the following preprocessing steps: drop columns, dummy variables, add quadratic interactions terms, standardize predictors

Solution: All those preprocessing were shown at 'assignment2_raquelaoki.py' with comments. Only two variables were nominal and we had to transform it into Dummy variables: country group and position. After add the quadratic interactions terms, the dataset went from 22 features to 253. Then it was excluded the interactions between two dummy features. The standardization was made before split the dataset into training and testing set.

2) Evaluating a weight vector: Write code that takes as input a weight vector and outputs the squared-error loss (see text) that results from predicting sum_7yr_GP using the weight vector.

Solution: In this question, it was develop a function called 'evaluation()' that receives as parameters the weight vector, the dataset and the lambda value as input and returns the square-error loss. To check the its implementation, it was created a unit testing ('test_evaluation') that receive as parameter a lambda and it use a simple value of weight and dataset to test the function 'evaluation()'.

It was used the Equation 3 to solve this question.

3) Finding a weight vector. Write code that takes as input a regularization parameter and outputs (i) the optimal weight vector as defined in the previous theory exercise (ii) the squared-error loss for this weight vector.

Solution: In this question it was used the result shown in Equation 4. The function developed in Python and it is called 'finding_weight()'. This function receives as input a dataset and a lambda value, and returns a vector with the weighs. The function 'test_finding_weight' receive as input a lambda value and test the function 'finding_weight()' in a toy example.

Grid Search: Now you have working code to perform linear regression. The main issue is finding a good value for the regularization parameter . Let us try an exponential grid search, as follows:
1) Try the values from the set $\lambda = 0.01, 0.1, 1, 10.100, 1000$. Make a plot that shows the following. The horizontal axis shows the value of on a log scale (this is called a semilogx plot). One curve in the plot should show the squared-error loss evaluated by using 10-fold cross-validation on the training set. The second curve shows the squared-error loss evaluated by applying the learned weight vector to the test set. Put this plot in your report, and note which regularizer value you would choose from the cross-validation, and which regularizer value would give the lowest squared-error on the test set.

Solution: The complete code for this question is shown at $assignment2\_raquelaoki.py$. First, the training set with the years 2004, 2005 and 2006 were randomized and split in 10 parts. Thus, the cross-validation were performed 10 times and in each interation 1 of these 10 parts was used as testing set and the others 9 as training set. Each element was used in the testing set exactly once.

The lambdas proposed were $\lambda = 0, 0.01, 0.1, 1, 10.100, 1000$, but it was implemented $\lambda = 0.01, 0.1, 1, 10.100, 1000, 10000, 100000$. These changes were made because the $\lambda = 0$ presented values worse than the others, so it was excluded; $\lambda = 10000, 100000$ were add because at $\lambda = 1000$ the squared-error loss was in its minimum. After these changes, it was possible notice in Figure 2 that $\lambda = 1000$ is the best regularizer, because it has the lowest squared-error loss in the testing set.

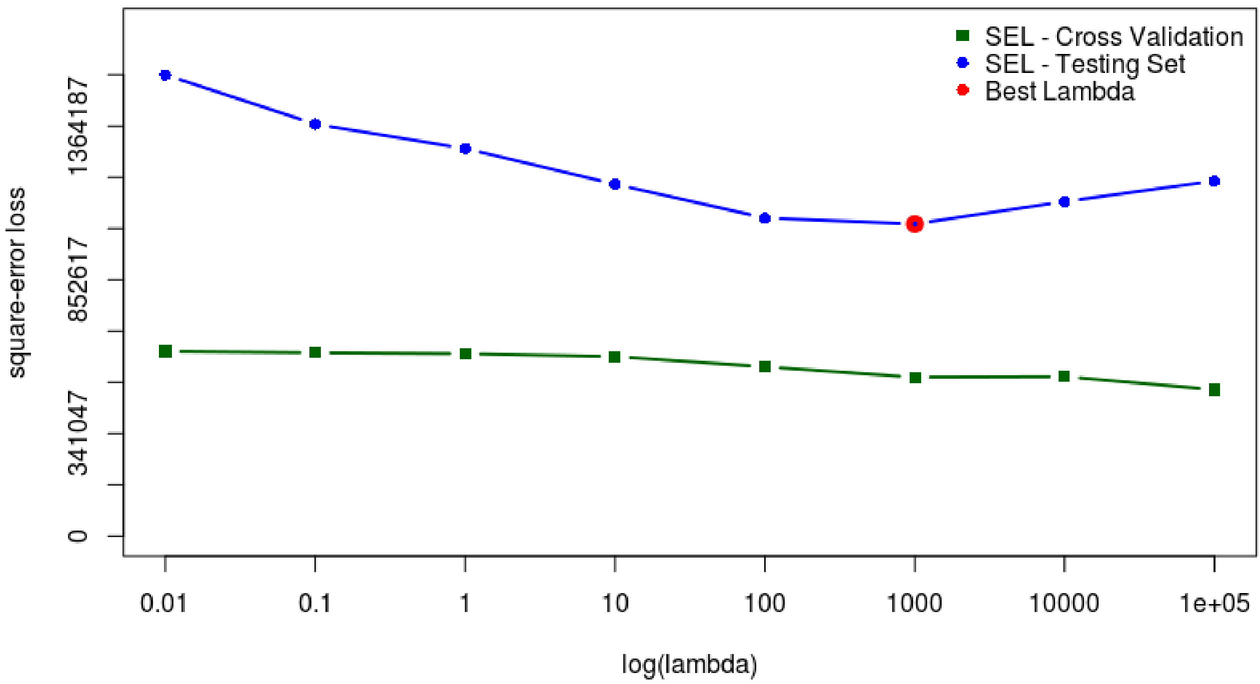

Figure 2: Grid Search. Best lambda is 1000

2) For the regularizer that you chose as best from cross-validation, inspect the learned weight magnitudes. Are any of the quadratic interaction terms important (i.e. carry significant weight compared to other variables)? The decision tree also captures interactions among predictor variables - how do the decision tree interactions compare to the interaction terms with high weights?

The Table 3 shows the Top 20 Features with the largest absolute values of weight for $\lambda = 1000$. Many interactions are relevant, such as DraftAge-Weight, CSS_rank-rs_G, CSS_rank-rs_P, Height-Weight , Weight-CSS_rank. In the decision trees, if a node is split with CSS_rank and one of its child is split with rs_G, it means that there is a interaction between these two features. The decision tree important interaction terms are the features on the top and are $CSS\_rank + rs\_G$, $CSS\_rank + rs\_P$ (Listing 1).

These quadractic terms are also relevant in the Regression, once that they are on the top 5 features with largest absolute weights (Table 3). So, those terms are indeed relevant for this problem.

Table 3: Top 20 features with the largest absolute weight

Feature	Weight
DraftAge	5.038185
Weight	4.457740
CSS_rank+rs_G	4.431261
CSS_rank+rs_P	3.415605
Weight+rs_A	2.691523
Weight+CSS_rank	2.679514
Height+CSS_rank	2.543294
CSS_rank+rs_A	2.423021
rs_PlusMinus+country_group_EURO	2.414067
Weight+rs_P	2.336839
CSS_rank+country_group_USA	2.336583
po_GP+Position_R	2.334206
Height	2.295350
CSS_rank	2.282933
rs_PIM+country_group_EURO	2.244322
po_A+Position_L	2.218636
CSS_rank+country_group_CAN	2.206787
CSS_rank+rs_PlusMinus	2.008091
country_group_USA+Position_R	1.955599
CSS_rank+Position_L	1.951842