JMLR: Workshop and Conference Proceedings 1–14, 2016 Submitted 4/23/2016

Estimating Soccer Player Performance with Similarity Search Regression

Max Henry MHENRY22@JHU.EDU *Johns Hopkins University*

As more and more money floods into the game of international soccer, the margin for error given to managers and scouts gets smaller. As a result, the pressure for instant success in the modern game is the highest it has every been. Unfortunately, the success rate for identifying players that can help a team improve is low. As a first step to address this issue, I propose a k-nearest neighbor regression model that performs similarity search on a qualitative set of attributes that describe "how" a player plays the game. The results of the similarity search are used to estimate the impact a player should expect to have for their team, as measured by their expected goal output for one season. In order to show its effectiveness, this model is compared to two linear regressions algorithms and outperforms both in a statistically significant way. **Keywords:** Machine learning, k-nearest neighbor regression, similarity search

1. Introduction

In comparison to baseball, soccer's data revolution has been late to start and even later to pick up momentum. However, seeing the success that smaller market teams in Major League Baseball have had, analytics in the "Beautiful Game" is a area owners are exploring as they try to compete against teams with bigger budgets [5]. Since the early 1990s, the money in and around the top European leagues has increased to levels never before imagined by the sport and, as a result, the pressure for a club, from the front office to the management and down to the players, to perform has never been higher [6]. Evidence of these big sums of money is most evident in the amount of money teams are paying for players. The world record for a player transfer fee in 1992 was 13 million English pounds. In 2016, the record sits at 85.3 million English pounds. This makes for worried reading for a professional scout, as their mistakes for endorsing the wrong player can no longer be covered up by a cheap transfer fee. In addition, because of the amount of money involved, managers are expected to get the most out of every player they bring in as quickly as possible. With that being said, the current success rate of scouting is dire: according to Paul Tomkins, only 40% of all players brought into a club go on to make a meaningful contributions to their teams. As a result, it is clear professional soccer scouting would benefit from a system that could identify the right players to bring in, players that correctly fit a team's system and have the skills to compete at the level required.

As a new field, data analytics in soccer is still unsure on what data best captures a player's performance and influence on the game. Many of the top managers would argue evaluating a player is best done by watching "how" they play the game, which is not necessarily represented by their pass length or shot speed statistics. In addition, when fans and managers

argue about what players their club needs in order to win, it is not long before someone compares that need to a current player in the form of "a player that runs like..." or "a player with the intelligence of...". As such, player comparison is a large part of the scouting vernacular today. In order to bridge the gap between "old-school" intuition and "new-school" analytics, I propose a system that performs nearest neighbor searches on qualitative player attributes and uses the results to estimate the expected impact of a player, measured by estimated goals output. Specifically, my hypothesis is that, because goal production based on qualitative attributes is a non-linear concept, k-nearest neighbor regression will outperform linear regression and ridge regression.

2. Approach

Below, I outline the structure of the data used for this experiment, the hyperparameter tuning methods I used, and the steps in my experiment. I use mean squared error as the performance metric and it serves as the basis for hyperparameter tuning and my comparisons between algorithms.

2.1. Data Sets

An additional result of the soccer analytics field being only newly established is that data repositories are few and far between and what does exist is owned by companies like Opta and Prozone. The main customers of these analytics companies are soccer clubs, not the public, and, as a result, their fees are prohibitively expensive for individuals. With that being said, it is also still an open question on how to use the mountains of performance data they have stored and it is unclear if this freedom would complicate my problem of estimating goal production from "how" players play the game.

As a result, I have taken advantage of qualitative data captured by EA Sports, which serves as the basis for their popular video game franchise "FIFA". This data set is specifically made public to complement a game mode that allows gamers to build fictional teams by mixing together players that do not play on the same team in reality while being constrained by team chemistry, the availability of each player, and cost. Since this is an active game mode, EA Sports actively maintains the data set to include every player in more than 25 leagues around the world.

An individual instance in their data set describes a player in 36 attributes. These attributes are meant to capture all of what determines a player's success during a game so higher scores are used for more successful players. Each attribute can range from zero to 100. A list of the attributes and a small description for the less obvious attributes is given below. It is important to note these attributes generally describe "how" a player plays the game as opposed to "what" the player does during a game, as captured by, for example, average pass length or the number of shots on goal. This makes it easier to discuss in what ways two players are similar.

The data from EA Sports does not report goal production for any of the players included. For this data, the names of individual goal scores for each league captured in the EA Sports

Figure 1: EA Sports Data Attributes

Acceleration	
Sprint Speed	
Agility	
Balance	
Reactions	How long it takes a player to react to certain game events (i.e. an opposing player being out of position)
Ball Control	
Dribbling	
Positioning	
Finishing	How consistently a player scores when they shoot
Shot Power	
Long Shot Accuracy	
Volleys	How cleanly the player can kick the ball as it is coming out of the air
Penalty Kick Success	
Interceptions	
Heading Accuracy	
Short Passing Accuracy	
Long Passing Accuracy	
Curve	The amount of curl a player can apply to a shot
Jumping	
Stamina	
Strength	
Aggression	

data was scraped from transfermarket.co.uk. Only goal production for a single season, 2014-2015, was processed and added to the EA Sports data. The EA Sports data only contained last names, which are not unique enough to match the correct goal production as reported by transfermarket.co.uk. As a result, full names of all players had to be scraped from futwiz.com. In the end, a single data set containing the full names of every player, their ratings for each of EA Sports' 36 attributes and their goal production for the 2014-2015 season was compiled. As I am focused on goal scoring, I removed all goalkeepers, which left 12,365 players. This represents the complete data set used for all hyperparameter tuning and experimental testing.

2.2. Model

In this project, I am trying to predict the goal production of a player by comparing them to "similar" players and their goal production. In machine learning, this can be done by using k-nearest neighbor regression. A brief background is given in the "Background and Related Work" section of this paper. Specifically for this problem, I plan to represent each player in the EA Sports data set as a 36-dimensional vector with each dimension corresponding to one of the attributes listed in the figure above. When the model is presented an unseen example, it will find the k closest training examples to the query example and use the goal production of the corresponding players to generate a prediction for the goal production of the query example.

2.2.1. Hyperparameter Tuning

k-nearest neighbor regression models have a number of tuneable parameters that combine to influence its performance. Specifically, these include a distance metric used to determine the closest (most "similar") neighbors, the number of neighbors to consider when generating a prediction (k), and how to weigh each neighbor's contribution to the prediction. In addition, according to [8], vectors of more than 20 attributes are considered high dimensional. High dimensional data has a number of effects on standard machine learning algorithms. As a result, dimensionality reduction techniques can be used to map the original data into a lower dimensional feature space where these harmful effects disappear.

The above concerns are taken into account in this experiment by first choosing an appropriate distance metric. During this investigation, I considered the Euclidean, Manhattan, Chebyshev, Canberra, and Brycurtis distance metrics. In addition, based on the work of [9][10][11][12], I consider two fractional distance metrics.

Next, I apply stepwise forward feature selection and stepwise backward feature elimination, in hopes of reducing the dimensionality of the feature vectors from 36 so that less high-dimensional special treatment of the utilized algorithms is needed. Lastly, I run a few tests to determine the optimal k for the model and whether to weigh the contribution of each neighbor to the final prediction by distance. The results of this tuning process are discussed in the "Results" section. During hyperparameter tuning, I used 10-fold cross validation and follow a factorial study pattern, where once an optimal setting of a parameter is determined, that value is used for the tuning of other hyperparameters.

2.3. Comparison Experiment

After tuning the k-nearest neighbor model, I compare its performance to linear regression, ridge regression and radius nearest neighbor. Radius nearest neighbor models are similar to k-nearest neighbor regression models but instead of considering k neighbors in the prediction, radius nearest neighbor only consider neighbors within some predefined radius of the query example. In order to fairly compare the results of the k-nearest neighbor model after the surprising success of stepwise forward feature selection, I run each of the comparison algorithms as the basis for stepwise forward selection and use the resulting feature set in the comparison only if it improves on the performance of the algorithm run on the full feature set.

2.4. Implementation

Sklearn is an open-source machine learning library written for Python and available from scikit-learn.org/stable/. All algorithms tested in this project use the sklearn implementation of the algorithm. The only algorithms that need implemented were SFS, stepwise backward elimination, and the fractional distance metrics.

3. Background and Related Work

3.1. k-Nearest Neighbors

k-nearest neighbor algorithms are instance-based learning methods that can be used for similarity searches, classification and regression. In general, instance-based learning methods are different than most others families of machine learning algorithms in that the training phase of these algorithms generally consists of simply storing the data. Then, when a query is put to the model, a group of similar instances are retrieved from memory and, in the case of regression, used to generate an estimate of the target value of the query [1]. Due to when the learning actually takes place, these methods are also known as lazy learning methods. k-nearest neighbor algorithms are simple examples of instance-based methods where data instances are represented as d-dimensional vectors and correspond to point in a d-dimensional space. These algorithms are used for similarity searches by retrieving the k nearest data instances to the query instance, as defined by some distance metric or similarity function. Common distance metrics include Manhattan distance, Euclidean distance, and max-coordinate distance [2].

3.2. Space Partitioning Data Structures

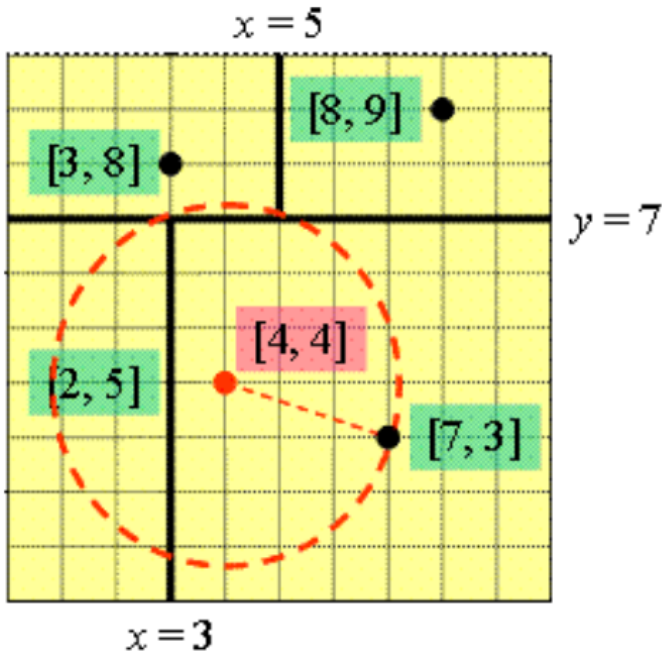
In order to efficiently search for the neighbors of a query instance, space partitioning data structures have been developed, of which kd-trees are a popular example. Kd-trees store the data instances from a d-dimensional space in a binary tree by splitting the instances on hyperplanes, creating hyperrectangles containing the instances [2]. In general, the main types of kd-trees differ on where they store the data instances, either on the interior nodes of the tree or on the leaves. With data instances as leaves, interior nodes represent tests on one dimension of the data instance vector and balanced trees of this form can, on average, be traversed in $O(d \log |P|)$ where P represents the set of data instances. When it comes time to query for a specific data instance's nearest neighbor, assuming two dimensions for simplicity, one firsts follows the tree down to its leaves by testing the query instance at each interior node. Once at a leaf, one has an initial approximation for a nearest neighbor. Once here, care must be taken to compare the initial result to instances in other partitioned spaces. As an example, the figure below shows five data instances defined in two dimensions. Assuming the query point is (4,4), the initial guess of nearest neighbor is (7,3), which is incorrect. As a result, the next step of the algorithm is to draw a circle with radius the size of the distance from the query point to the initial guess and investigate all partitioned-spaces that intersect that radius. If a closer point is found, a new circle drawn with a radius of the distance between the query point and the current nearest neighbor and the process is repeated [2].

Ball trees are similar to kd-trees but partition the space into hyperspheres instead of hyperrectangles. Because of the higher degree of freedom when testing distance metrics in sklearn, ball trees are used for all tests in this project.

3.3. Distance Metrics in High Dimensions

As mentioned in [12], two and three dimensional intuitions are warped more and more the higher the dimension. A consequence of this is standard Euclidean distance loses its meaning

Figure 2: Kd-tree Example



as the training examples lose the contrast between their nearest and furthest neighbors [9]. This loss of contrast can have a major effect on the number of branches that are searched in a space-partitioning data structure, thus possibly losing any perceived performance advantage by returning meaningless nearest neighbor queries. To combat this issue, researchers can try reducing the dimensionality of the data, which is tested here, define a distance metric that preserves the contrast of nearest-to-furthest neighbor, or repose the problem as outlined in [10][11]. It was found in [9] that fractional distance metrics (l_p norm, $p \in (0, 1)$, specifically p=0.3) performed well on synthetic data sets with dimensionality of 20. As a result, two fractional distance metrics, p=0.3 and p=0.1, are briefly tested during hyperparameter tuning.

4. Results

This section details the results of hyperparameter tuning and the comparison test between algorithms. In addition to the original data set, I considered a standardized and normalized version of my data set. In the standardized version, each attribute was shifted to have a mean of zero and a standard deviation of one. In the normalized version, each attribute was scaled down to fit in the range (0,1). These two data sets were created mainly for exploratory reasons as I was searching for ways to improve the performance of the k-nearest neighbor model. They performed worse when fed to any of the algorithms tested so their results are not reported here, besides for the distance metric tuning.

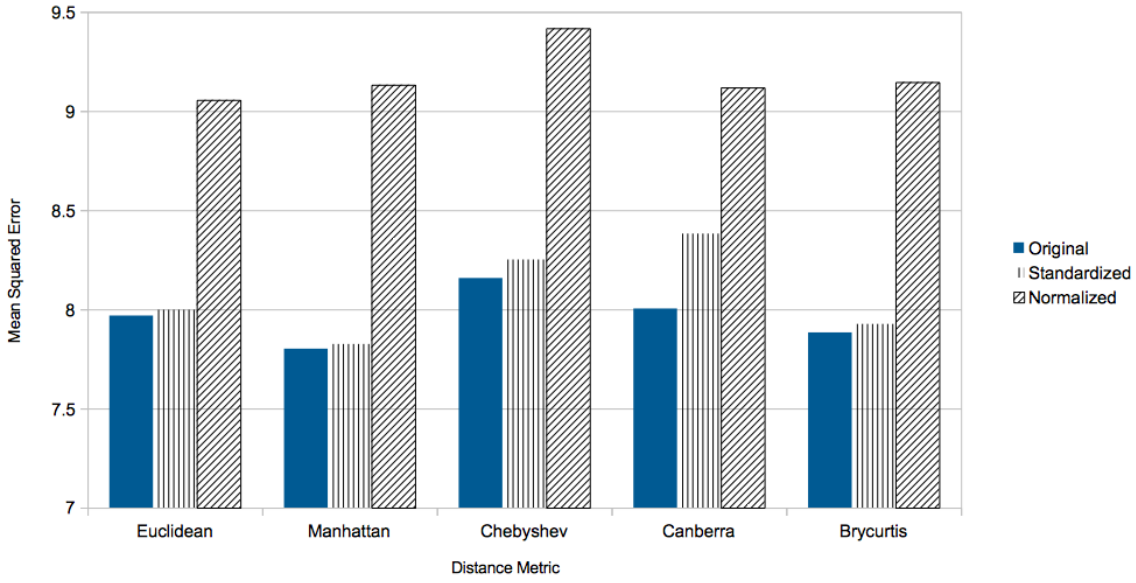
4.1. Hyperparameter Tuning

4.1.1. Distance Metric

Due to the high dimensionality of the original data, choosing a distance metric that created a meaningful spatial relationship between the data and the contrast between the nearest and furthest neighbor was my main concern. To this end, I ran the full feature set through 10-fold cross validation using the Euclidean, Chebyshev, Manhattan, Canberra, and Brycurtis distance metrics. Given the results of [9], I also tested two fractional norms, p = 1/3 and p = 1/10, however, the results in terms of mean squared error of their initial cross validation tests made them the weakest distance metrics tested. In addition, the runtime of the tests, due to a limitation of sklearn, with fractional metrics were such that multiple tests on the normalized and standardized data sets were not possible. As a result, the results of the fractional metrics are not reported here.

In addition, a normalized Euclidean distance metric as recommended by [13] is used to keep an attribute with a wide observed range from dominating the distance metric calculates. However, in the case of the original EA Sports data set, the mean range is 78.37 with a standard deviation of 7.57 and, as a result, I did not feel the need to compensate for those ranges by testing the normalized Euclidean distance.

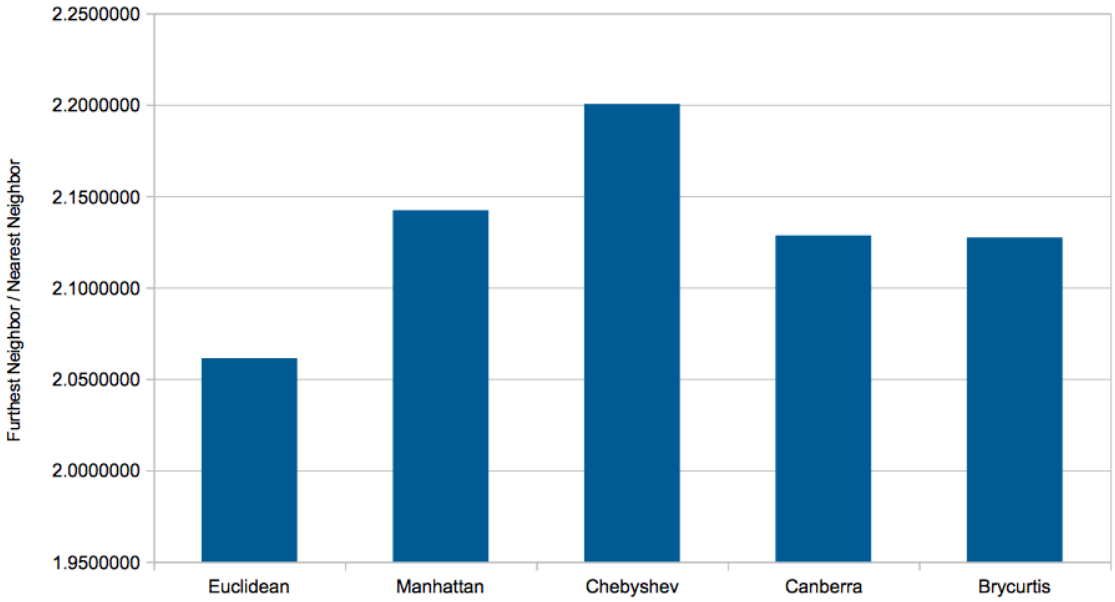
Figure 3: Distance Metric Performance on Full Feature Set



The figure above reports the performance of each of these distance metrics prior to feature selection. It also shows how the different data sets performed with each distance metric. It is apparent the normalized data set performed the worst, followed by the standardized data set and the original data set. Since the original data set performed the best with each metric, the normalized and standardized data sets were not used in later testing. For the original data set, it is clear all five metrics performed about the same and their differences

are not statistically significant according to ANOVA testing. In addition, the figure below shows they all preserve the contrast between the nearest and furthest neighbor the same. This ratio was used by [9] to show the breakdown of certain distance metrics as p increases in the l_p metric. I use this same metric to show that all the tested distance metrics here appropriately contrast near and far neighbors. As a result, the rest of the hyperparameter tuning proceeded with the Manhattan distance metric.

Figure 4: Ratios of Furthest Neighbor / Nearest Neighbor Distance



4.1.2. Feature Selection and k

As mentioned in [9], when facing high dimensional data, researchers can 1) use dimensionality reduction on the feature set 2) develop a new distance metric that operates in higher dimensions or 3) reformulate the nearest neighbor problem. Since the fractional metrics did so poorly during distance metric tuning, the need to reduce the dimensionality of the original data became important. In order to do this, I tested stepwise forward feature selection and stepwise backward elimination. In addition, I chose to combine the features selected by forward selection with a set of attributes I determined as an "expert". The idea was that I might be able to counter how forward selection breaks up groups of attributes that only perform well together by reintroducing a feature group I deemed as relevant.

I noticed during testing that the performance of the k-nearest neighbor model when run on the returned feature sets from forward selection and backward elimination varied as k varied. As a result, I decided to test these two hyperparameters together. For each feature set, I ran 10-fold cross validation for a number of different k values and the results are displayed below. Stepwise backward feature elimination took longer to run so fewer data points are presented.

Figure 5: Stepwise Forward Feature Selection

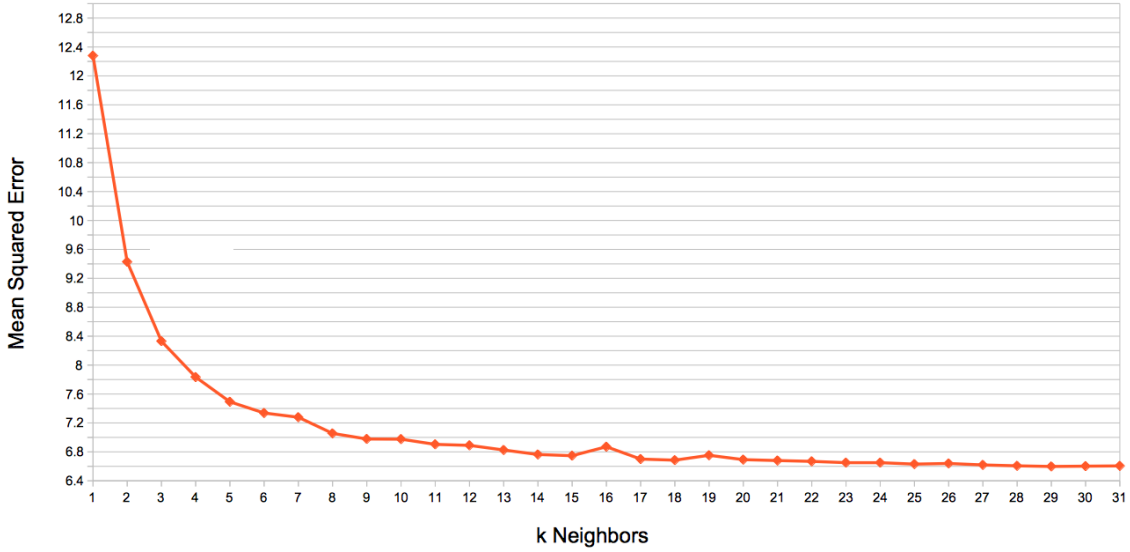
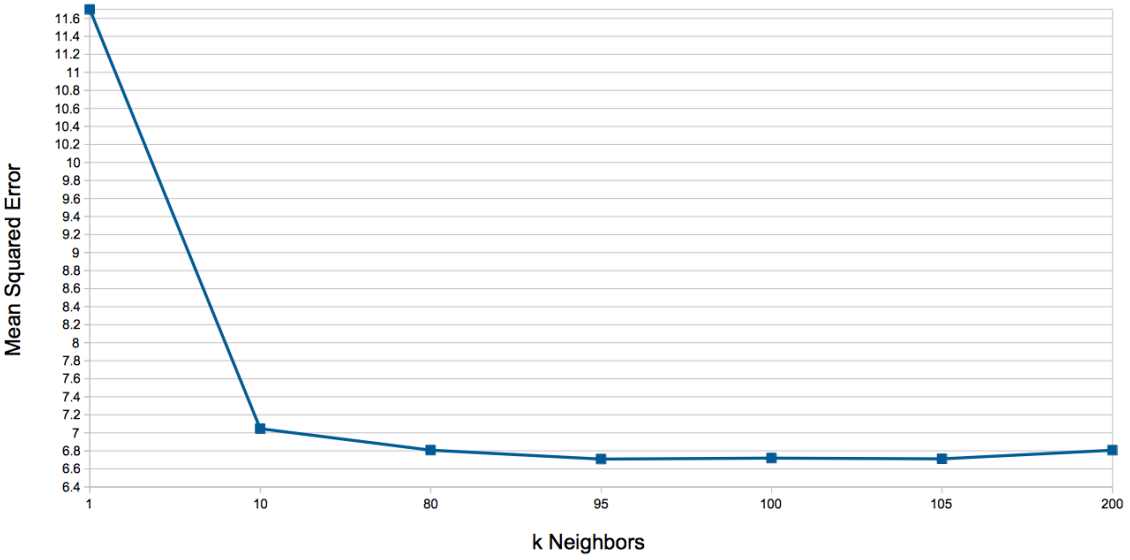


Figure 6: Stepwise Backward Feature Elimination



Forward selection performed best at $k = 29$ and backward elimination performed best at $k = 95$. Stepwise forward selection returned only five of the 36 attributes ("Finishing", "Reactions", "Positioning", "Stamina", "Dribbling") with a MSE of 6.5982. Stepwise backward elimination returned six attributes ("Reactions", "Ball Control", "Positioning", "Finishing", "Short Passing", "Stamina") with a MSE of 6.7084. In addition to the features forward selection returned, I added "Shot Power", "Long Shot Accuracy", "Volleys", "Penalty Accuracy", "Free Kick Accuracy", and "Curve". All of these attributes represent different ways a player can score, but combining them with the forward selection features

actually hurt performance (MSE = 6.8937). These results are discussed in the "Analysis" section. Because of the better performance and fewer neighbors needed, I chose the feature set returned by stepwise forward selection for the final model.

4.1.3. Neighbor Weighting

The last hyperparameter to tune was to determine how neighbors would contribute to the query example's predicted goal production. Weighing neighbor contributions by the inverse of their distance from the query example for $k = 29$ performed slightly poorer than apply uniform weights to each neighbor's contribution. Because of this, uniform weights were chosen.

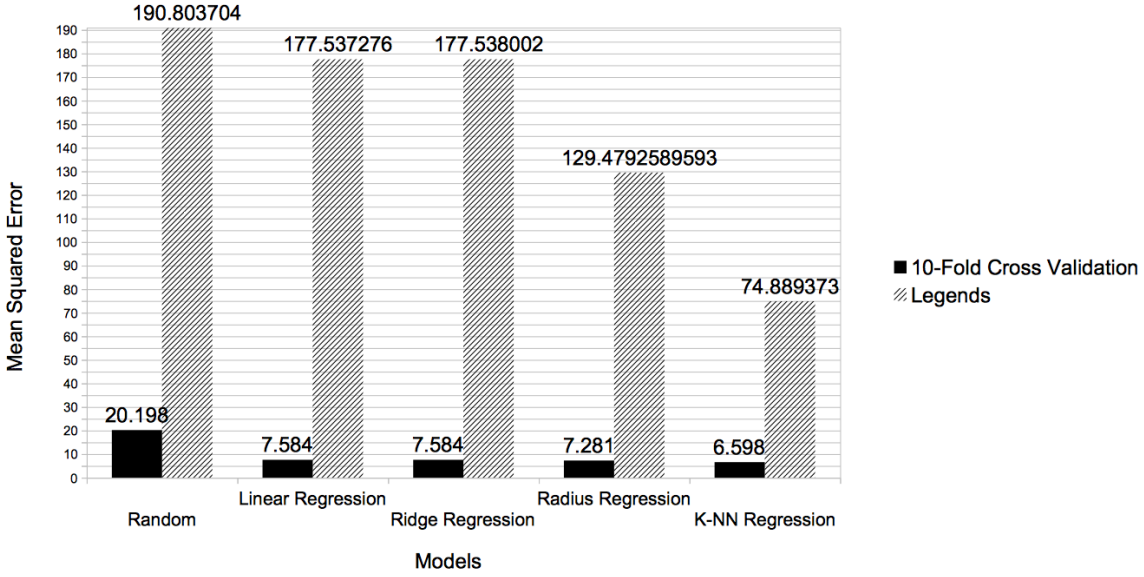
4.2. Algorithm Comparison

For the final algorithm comparison, I trained all models on the full EA Sports data set and goal tallies. I tested generalization performance with a set of 54 "Legendary" players that do not exist in my subset of the EA Sports data set. As all of these "Legendary" players are retired and do not have goal tallies for the 2014/2015 season, I assigned them based on the highest number of league goals they scored in their careers for a single season. As a result, this test set is noisier than the goal tallies in the training data, which are derived from the 2014/2015 league season.

I compared four algorithms including a linear regression model, ridge regression model, a radius regression model and my k-nearest neighbor model. In addition, in order to provide a baseline, I included a model that output random goal tallies for predictions based on the distribution of the goal tallies of the players in the training set. I ran the configuration for each algorithm that performed the best on the EA Sports data set during 10-fold cross validation. Each algorithm's performance in terms of MSE was averaged over 100 runs. The results are presented below.

As can be seen, the k-nearest neighbor model outperforms the linear regression models on both the cross validation tests and the "Legends" data set. The performance seen by the k-nearest neighbor model is particularly good for the cross validation set with a MSE of 6.598. This translates to an average error of 2.5 goals per player. I claim this margin of error is good and allows all the predictions made by the model to be used by professional scouts. It is important to note that a system designed to fill the needs of a professional scout does not need a margin of error much smaller than 2.5 goals per player as the predictions outputted by the model will serve as a first step, rather than a final decision, in assessing a players utility to a new team. The k-nearest neighbor model's dominance is particularly evident when generalizing on the "Legends" data set and possible reasons for this are discussed in the "Analysis" section. For both the cross-validation and "Legends" data set, the performance differences between the k-nearest neighbor model and the linear and ridge regression models are statistically significant. This confirms my hypothesis.

Figure 7: Comparison of Non-linear and Linear Regression Models



5. Analysis

5.1. Hyperparameter Tuning

As mentioned above, based on the contrasts between the nearest and furthest neighbor as determined by the tested distance metrics, contrast was not lost for any tested metrics. It is clear in the literature of [8] and [10] that any dimension above 20 is when typical distance intuitions breakdown and the term "nearest neighbor" may begin to lose meaning. A possible explanation for this is that, for example, Euclidean distance assumes each dimension is equally relevant so if a portion of the high dimensions contain noisy attributes, it is possible to deceive the metric. [9][10][11] do not completely define what value of contrast shows a poorly functioning distance metric. As a result, it is possible the values around two that are observed for the EA Sports data set means the metrics have already broken down, but it should be noted that [9] and [10] both include data sets whose maximum contrast as defined by the author's fractional metrics have values of two. It should also be noted that for the Manhattan metric, contrasted increased to 5.4 when the subset of features identified by stepwise forward selection were taken into consideration. This would seem to suggest that the best option for dealing with high dimensional data is to apply feature selection in hopes that a higher performing subset of features can be identified.

Both stepwise forward selection and stepwise backward elimination pruned at least 70% of the features in the data. It was clear that some attributes like "Marking", "Interceptions", and "Sliding Tackle" did not relate to goal scoring while it was unclear if other attributes like "Acceleration" or "Vision" would be useful. It was however, surprising that attributes that seem directly related to goal scoring were not selected. This was the basis for me to try the features selected by stepwise forward selection with features chosen by an "expert".

The surprising result here is that including those attributes degraded performance. By selecting one attribute at a time, stepwise feature selection may break up attributes that improve regression performance but individually are not useful. As a result, even with the "expert" feature subset some features might have been pulled from their higher performing group. It is unclear to me to what larger group the "expert" features could belong so, especially considering stepwise forward selection improved model performance the most of any hyperparameter, a more critical investigation of the optimal feature set is warranted. Of the features selected by stepwise forward selection, "Reactions" is the least clearly relevant feature. This may indicate a feature that was plucked from a higher performing group.

Another surprising result of tuning the k-nearest neighbor model is how well using uniform weights to determine each neighbors contribution to the final prediction stood up to the inverse distance weights. This indicates that players are not cleanly clustered with similar players all equidistant from them. It should be noted the difference in performance between the uniform and inverse distance weights was small, which means that, although there are not clean clusters, the clusters that do exist are not spread out over a large distance. I, unsuccessfully, tried applying different functions to determine the weigh for each neighbor's contribution that was based solely on the number of neighbors that have a non-zero goal production but was never able to improve upon uniform weights. Future work may further investigate new ways to determine weights.

As noted above, the k-nearest neighbor model reached optimal performance when it considered 29 neighbors. Such a number, as opposed to the 95 neighbors needed for the stepwise backward elimination feature set to reach optimal performance, suggests "subtypes" of players exist. This fits with conventional wisdom in soccer, where "roles" have been defined that describe how players play. These "roles" including descriptions like "playmaker", "poacher" (referring to the goal scorer who only needs one shot to score), and "flashy" (referring to players that use tricks to beat their defender). It is unclear if these 29 neighbors correspond to these roles but it may make for interesting future work.

5.2. Algorithm Comparison

In addition to showing how the k-nearest neighbor outperforms linear regression, testing with the "Legends" data set may have indicated how sensitive linear regression is to outliers and noisy data. The "Legends" data set is a set of players that may be considered outliers because, on average, their goal-scoring-related attributes are much higher than the average player in the EA Sports data set. As a result, the neighbors for the "Legends" are further away. In addition, their goal tallies are their best goal production in a league season over their entire career. This inflated their goal production and, for future work, it would be interesting to see what the effect on performance is if the goal tallies are replaced with their average league season goal production. Although an error of 8 goals per player is high for the k-nearest neighbor, considering the characteristics of the "Legends" data set, it is reasonable.

The 2.5 goal error per player that the optimal k-nearest neighbor model returned for the cross validation test is reasonable. Essentially any player healthy for an entire season is capable of scoring two goals, regardless of position and skill set (assuming they can compete at the league's skill level). A lower error per player would be better but, as far as applying a regression algorithm to a novel data set goes, the current level is acceptable.

6. Future Work

Due to the success of the feature set returned by stepwise forward selection for the k-nearest neighbor regression model, a first step the soccer data analytics community could make in discovering relevant performance statistics is to try and quantify the features in that set. As a reminder, this feature set included the "Reactions", "Positioning", "Finishing", "Stamina", and "Dribbling" attributes. Specifically, it is unclear how to capture the "Reactions", "Positioning", and "Dribbling" attributes. If quantified statistics can be captured, it would be interesting to see how they affect the performance of a k-nearest neighbor goal predictor.

In addition, the findings of [9] may need to be probed as to how much using synthetic data sets influenced the results the authors reported. In my case, 36 attributes did not effect the contrast between the nearest and furthest neighbors each "typical" distance metric created. As a result, the discussion of how many attributes make a data set high dimensional may be better posed as what specific data characteristics cause the "typical" distance metrics to break down.

Two basic feature selection algorithms were used to determine a quality subset of features to use in the k-nearest neighbor model. The stepwise forward selection algorithm introduced the greatest improvement in terms of MSE during 10-fold cross validation. As a result, it would be interesting to see what effect a "smarter" feature selection algorithm would have on the MSE of the model, such as PCA.

7. References

[1] Mitchell, Tom M. Machine Learning. New York: McGraw-Hill, 1997. Print.

[2] John Hopkins Unversity Engineering for Professionals, EN.605.746.81.SP16 Machine Learning, Dr. Sheppard, "Instance-based Classification", p. 93

[3] Yingying, L. I., Silvia Chiusano, and Vincenzo DElia. "Modeling athlete performance using clustering techniques." The Third International Symposium on Electronic Commerce and Security Workshops (ISECS 2010). 2010.

[4] "Hilltop Analytics." Hilltop Analytics. Web. 13 Mar. 2016. ¡http://www.hilltop-analytics.com/football/find-me-a-player-like-andres-iniesta/

[5] White, D. (n.d.). Liverpool owners looking to use baseball principles of statistical analysis. Retrieved March 13, 2016

[6] Gibson, O. (2015). Sky and BT retain Premier League TV rights for record 5.14bn. Retrieved March 13, 2016

[7] Tomkins' Law: Only 40% of Transfers Succeed. (2014). Retrieved March 13, 2016, from https://tomkinstimes.com/2014/06/tomkins-law-only-40-of-transfers-succeed/

[8] Berkhin, Pavel. "A survey of clustering data mining techniques." Grouping multidimensional data. Springer Berlin Heidelberg, 2006. 25-71.

[9] Aggarwal, Charu C., Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. Springer Berlin Heidelberg, 2001.

[10] Hinneburg, Alexander, Charu C. Aggarwal, and Daniel A. Keim. "What is the nearest neighbor in high dimensional spaces?." (2000).

[11] Aggarwal, Charu C. "Re-designing distance functions and distance-based applications for high dimensional data." ACM Sigmod Record 30.1 (2001): 13-18.

[12] Domingos, Pedro. "A few useful things to know about machine learning." Communications of the ACM 55.10 (2012): 78-87.

[13] Bao, Yongguang, Naohiro Ishii, and Xiaoyong Du. "Combining multiple k-nearest neighbor classifiers using different distance functions." Intelligent Data Engineering and Automated LearningIDEAL 2004. Springer Berlin Heidelberg, 2004. 634-641.