**Imagine you are hired by a sports team, and you are asked to build a machine learning model to evaluate trades. I.e., you want to predict how players perform on a new team compared to their performance on their old team (n.b., think of soccer, basketball, football, baseball, etc. – team sports). Specify how you would design your model, what data you would need, if any statistical assumptions are violated with these data / how you’d address this, and whether or not you think you would be able to produce a “good” model. Finally, which sports would be hardest to evaluate trades? Which would be easiest? Why?**

**DV:**

My first step would be to identify a dependent variable; this DV would have to somehow quantify performance of a particular athlete. My initial thoughts were points scored within a particular sport, however, using this as a dependent variable is easily undermined by the fact that defensive players are not necessarily working to score points. Building upon the fact that this dependent variable should encompass all players within a particular sport, I thought about using market value. Market value is a useful representation of performance because as a player does well, it is intuitive that his market value will rise. However, market value is also heavily impacted by age, position, and other qualities of a player like popularity. Thus, too many variables impact market value for it to be indicative of performance. I argue that using something similar to fantasy points is an indicator of performance for a particular player since it represents a direct correlation between positive, or negative, impacts of a player and an increase, or decrease, in ‘fantasy’ points scored.

My DV of ‘fantasy’ points for a particular player, however, only represents the performance of said player, and not that of the team. The question is framed such that I am predicting a singular player’s performance for a possible trade, and not how well the team will do, and it is entirely possible for the addition of a new player to a team to cause a team’s overall performance to drop while the new player’s remains as a high level. That could be due to other factors such as coaching style, player support in the new team, changes in player role, and adjustment in a new environment, etc. Thus, I will ignore this possibility because my task is to solely predict a player’s performance on this new team.

**MODEL:**

I am essentially attempting to model an answer to whether or not a particular player is viable on my team while limited to information about a player’s performance on previous teams and the general performance of my own and other teams. Thus, it is possible that the only reason a player is doing well is because of his teammates. For if these teammates are taken away, say the player is traded, there is no guarantee that the player will be able to perform at his previous level. Since I am using data regarding a team sport, I have to be extremely cognizant of the importance of balance within a team. Thus, I am certain that this model would have to consider players surrounding the player in question of being traded.

To do so, I would like to quantify certain attributes of a Team which are linked to its playstyle. For instance, I could use measures like team offensive rating and team defensive rating. Using these measures as IV’s would be helpful in determining whether the player in question is a good fit for a team. Thus, I’d like to create a latent variable for a team’s offensive and defensive capabilities (or the equivalent of offense and defense for each respective sport). Additionally, I’d like to use a variable which measures how effective a team is when a particular player is playing. This type of statistic exists in basketball and is called Real Plus Minus. I imagine similar measures exists in other sports as they are essentially just a measure of how much a team outperforms the other team when a certain player is playing. This variable would be highly influenced by the surrounding players and allow for an insight into how a particular player impacts the team. I also considered simply adding performance measures of each of the players previous teammates, however, this does not quite give insight into how the player interacts with the teammates.

After identifying 3 team-related variables, it is also important to give a measure of individual performance. I argue that this is necessary to give more weight to heavily influential players. For instance, a player may play on a poor performing team, however, the player themselves may be an exceptional talent (Wembanyana on the Spurs). For this reason, it is necessarily to include variables that relate solely to player performance. A complication that occurs is that we must now account for different roles players occupy. Thus, I propose a model for each position within a specific sport and to include performance IV’s relevant to that position. For prediction of a striker position in soccer I’d use some measure of goals scored per game and other relevant scoring related statistics; whereas for a point guard I may measure assists per game or turnovers per game. Within positions, I’d also like to create latent spaces that group certain aspects of a position; this is a useful step because it minimizes multicollinearity in the data- a common issue in player data. I could have a general set of player performance variables that capture offensive, defensive, and playmaking performance normalized by playing time (PPG, RPG, APG in basketball); for each position type, I’d train a different model as the performance IV’s would have different predictive power depending on the position of the player. x

**DATA:**

The data that I’d like to use comes would come from analyzing team and player statistics, for a particular position for the previous 2 seasons (that seems appropriate, as player performance changes in time with age, experience, and there is the risk of data becoming nonlinear if I include too many seasons. This is because as you add more years, the likelihood of performance being affected but other factors like age and injury increases. Additionally, I mentioned earlier that I would try to create a latent space and that I would target variables that have a linear relationship with the dependent variable. Furthermore, I am operating under the assumption that all the player and team data represents a true observation about the world. I would collect as much data about as many players and teams so that I can train the models for different positions very well. Also, I would start by including all the statistics collected in that sport, and then try to build latent variables for player performance and for the team using the design I discussed in the modeling part.

As for whether the data is IID, there are important points to make. I will assume the data is identically distributed because human behavior typically is normally distributed, and I don’t predict sports to be an outlier to this phenomenon (even though some players are at a significantly higher level than others, there are very few of these players). However, there could be an issue with independence of the data because every team has to balance offense and defense, or other aspects of the sport, and this could create a dependency between offensive and defensive variables (a team has to focus more on a particular aspect of the game which leaves less for other aspects). In this case, there would likely be collinearity within the data. Thus, I can work around this by creating latent variables of groups of IV’s that can ideally reduce multicollinearity.

Additionally, for top players in a sport, there could be a lot of variation because. For instance, if you were to plot data for a particular skill, most players would be at an average level and have similar variance, but at the top, some top players may perform extremely well and create much more variance; this could lead to heteroskedasticity with lots of variance at the end of the range from a particular skill of a group of players. To possibly circumvent this, I could rely more on a decision tree, or weighted regression, that can more easily capture the non-linear performance of top players.

**REGRESSION FORM:**

Regarding my previously chosen IV’s, I would like to input each variable into a regression to determine if they, in fact, are strong predictors of how well a player would do on a particular team. If it is the case that my regression gives an equation (or splits on) many variables that are team related, this will tell me that team dynamics are a useful indicator of how well a player will do.

For the linear regression, I would start with a lasso (this would allow me to see which of the IV’s are most important and can best predict player performance). However, to lower the amount of IV’s before I run a regression, I’d run a PCA on player data grouped by its role in the sport (offensive, defensive, supportive) and create latent variables for each subcategory of player data. After running a lasso regression, I’d also like to experiment with a decision tree as it will give me a good prediction of performance and another way to understand how the variables relate to performance. Additionally, this may allow me to capture certain non-linear relationships within the player data that may have slipped in regression.

I am straying away from black box models because, at least in the sports world, it is necessary to provide reasoning as to why a player might do well on a particular team to be able to communicate these results to a coach who may not have a background in modeling or statistics. However, I would run something like xgboost to allow for an upper limit on what my predictive power should look like.

In all, I believe a decision tree would be the best type of regression as it will allow the model to split between certain intricacies of team playstyle. For instance, say that the first split is on an offensive/defensive (based on position) latent variable, which would make sense since this is a good indicator of performance. Then, the decision tree would likely start to make splits on team related variables. For instance, if a team is below a certain threshold in defensive play, the tree would ideally learn that a defensive player would strive in such an environment and this split would lead to an increase in the final DV. The same is true for other possible qualities a team and player may mutually benefit from. In the end, the resulting tree would allow the coach to determine how well a player may or may not fit into the team based on his own team’s statistics and how well the player complements them.

**EVALUATION OF MY MODEL:**

Overall, I believe the aptitude of my model depends on the results of the decision tree/regression because if splits occur on team-related variables, I will know that these variables are important in predicting the performance of a player. This will affirm my initial hypothesis that a player will likely perform differently on a new team. However, if only variables that relate to player performance are chosen, this will tell me that team play is irrelevant to predicting a player’s viability on a trade. This is very likely not true as the theory and decisions regarding player trades are very nuanced and are not simply based off of which player the best.

To make sure I don’t overfit, I’d like to use cross-validation on the data and for testing data I could simply use the following season which contains players that have traded to another team. Other ways I reduce overfitting are by implementing k-fold which random partitions the data into k amount of training and testing folds for an unbiased evaluation of model performance.

Furthermore, the model of player performance does not consider the prices of players. I feel that this would offer a more realistic method to consider whether or not a player should be traded or not. To account for this, I would have to add a variable related to the market value of a player and ideally, I would only regress on players within a certain budget of the team.

**PERFORMANCE ON VARIOUS SPORTS:**

Amongst team-sports, I believe baseball is the easiest to evaluate because, according to my baseball research, a player performance following a trade in baseball is almost entirely based on individual performance. For instance, a player can make an immediate impact on a team simply because they are a good player (without having to learn any plays or develop chemistry). However, I mentioned earlier that my model is built off of using offensive/defensive data; this would have to be slightly altered for baseball since the roles of offense and defense are a bit different from that in other sports like soccer and basketball. I would imagine that my regression equation for baseball would show that team related variables are not particularly correlated with player performance. On the harder to model end of the spectrum, are sports like soccer and basketball where team play is crucial to player performance. This makes these sports harder to model because team related variables are harder to predict by nature (even after regression, there is no guarantee that a player’s particular playstyle will be implemented well by a coach/team). On the other hand, gathering data in the NBA and in soccer is, I assume, arguably easier than in other sports because machine learning technique are quite ubiquitous among teams, like the Celtics for instance; thus, there is already a strong incentive to measure and collect data. Additionally, football would also be quite hard to model considering teams all run relatively complex plays and a single weak player can hamper an entire team, regardless of the strength of the other players. Also, football is likely the most injury prevalent sport; this would make data gathering quite hard as many top-level players could have their stats significantly drop, at least more so than other sports.

**For a given sample, you start with a purely linear regression model and then try a polynomial regression of order 3. In both cases, model fit is similar using MSE and both models show heteroskedasticity. Which models would you prefer, all else equal? Given the presence of heteroskedastic errors, what do you assume is wrong with your approach? How can you correct this problem? Finally, let’s say you try a decision tree approach and MSE improves dramatically. What would you infer about the relationship of y ~ f(x)?**

In this case, I would prefer the purely linear regression to the polynomial regression of order 3, considering similar MSE and heteroskedasticity present. I prefer the linear regression because it has less capacity to overfit. Namely, I presume the polynomial regression will more easily fit to the heteroskedastic noise since its ability to account for higher order patterns in the data could easily misfit to noise. On the other hand, the simplicity of a linear regression offers a more rigid structure. That is not to say that the linear regression is impervious to heteroskedasticity, however, I argue that a polynomial regression model which fits to the noise of heteroskedastic more heavily hurts the predictive power of the model due to overfitting.

If I want to continue with the approach of linear regression, which I previously deemed to be more effective than polynomial regression, I have to somehow account for the presence of heteroskedasticity. First, I believe that the problem with this approach is that linear regression, which only attempts to minimize errors, gives equally weights each data point. This is an issue because heteroskedasticity implies that the variance between errors is uneven. Thus, when the regression happens, observations with lower variance will be treated the same as observations with much higher variance; the resulting issue is that the model will not be able to identify patterns in low variance observations versus unhelpful patterns in noisy high variance observations. A way to avoid this issue is to manipulate errors by weighting them differently such that there aren’t data points with much higher variance than others. This approach would allow the regression to not view datapoints with a higher error and assume that this error is meaningful. Although, this approach assumes there is a linear relationship within the data.

It is also possible that the regression model is not well specified and that some variables are not included. For example, after a variable reduction step, I could naively drop too many variables. Changes due to this ignored variable can be mistaken for changes in variance. One way to reduce heteroskedasticity would then be to revisit feature selection. It would also be beneficial to attempt running regression with an interaction variable if you are able to identify a quality of the data that gives it heteroskedasticity.

Now, if I decided to switch to using a decision tree and my MSE improves, I can assume that the decision tree is able to identify a way to split the data such that the heteroskedasticity in the data is reduced. For instance, in the example of modelling income vs education, STEM and non-STEM observations tend to display different patterns long-term, leading to heteroskedasticity. If the decision tree is able to first split on STEM vs non-STEM and reduce the most amount of variance, the rest of the tree will be able to form without having to somehow balance STEM and non-STEM datapoints together. If the decision tree is run with pruning, it would not miss to include a variable that could reduce error. On the other hand, the regression, with presumably fewer independent variables, would show increased and unequal variance. In this case **y**would be a function of more **IV’s** and a decision tree with pruning would do better. Another possibility is that the dependent variable does not vary linearly with the independent variables. Decision trees have the advantage of capturing nonlinear relation, unlike linear regression, which would explain a low MSE. Although, this is not pure heteroskedasticity, but it would appear as such due to non-linearity of the data. Hence, in this case, y is not a linear function of x.

**Explain equation 8.4 in the text. Relate this to the lasso, discussing similarities and differences. And detail how the particular value of alpha is chosen such that the model will not be overfit to any particular value of alpha.**

Pruning is an essential step in the formation of a decision tree because the greedy nature of this model would otherwise overfit a dataset. This is because the algorithm it uses simply attempts to minimize variance at every split and if this was left to run without penalty, it would result in an extremely complex decision tree that entirely overfits on training data with likely poor performance on a test set. Thus, there must be some method to prevent this from happening; this method is called pruning and is determined by equation 8.4.

Equation 8.4 has two parts:

1) a squared error part which is calculated in each of the regions partitioned by the leaves of the decision tree and then summed up over all leaves/regions.

2) a penalty part that is proportional to the number of leaves in the DT., this penalty works against trees which contain too many leaves to prevent overfitting.

This equation is utilized in cost complexity pruning. This process starts and the leaves with a fully grown tree and zero penalty (alpha=0) and successively increases the alpha to add more penalty for the number of leaves left in the tree. For each alpha, there is one tree that has the minimum cost complexity, as described by the equation 8.4. By using k-fold validation with the pruned trees as a function of alpha, the algorithm selects the best alpha such that it minimizes the average error.

This process is similar to equation 6.7 describing the error and penalty in lasso model. Regarding equation 6.7 for lasso, the predicted value (y hat) is obtained with a linear regression over independent variables, and the penalty part uses the absolute value of regression coefficients rather than number of tree leaves. Essentially, both models attempt to simplify a model where pruning reduces complexity within a tree and lasso sends coefficients to 0 if they do not have a relevant relationship with the target variable (lasso uses lambda while pruning uses alpha). Additionally, although both of these methods achieve similar goals, lasso implements regularization by summing abs. value of coefficients as a penalty whereas pruning essentially measures a tree’s complexity and uses alpha to determine the desired tradeoff between complexity and overfitting (it removes leaves accordingly). Within supervised learning, one can implement a search method in which a loop tries different lambda values with k-fold testing and selects the lambda with the minimum cost.

Considering equation 8.4, it is important to guard against overfitting since the algorithm essentially picks a certain value that produces the best result (in terms of MCC) with the given training data. However, as seen in algorithm 8.1, the model is created to not overfit since the cost is calculated as an average over the k-folds and the simplest tree that minimizes the RSS plus the penalty is chosen. This makes it costly to choose large, unpruned trees (which likely overfit) and also minimizes the fitting error. The model also tries over a number of alphas to select the best tree, so that many pruned decision trees are evaluated.