I. Team info and contributions

Team members:
Jasmine Harris
Edmond Anderson
Chris Grandy
Suliah Apatira
Sarah Bradford

Concerning the separation of project responsibilities: It is difficult to say who performed what part of our project as it was mostly completed in a group format. We had weekly meetings to work on the progression of our project, this is forum in which most of the work was completed.

II. Problem statement

In the NBA contracts hold a significant amount of power. Once a player is signed it is very difficult for a team to break the contract. In the past ten years many NBA front offices have saddled their team with albatross contracts with players that do not create the desired output. Signing the wrong player to a long-term high value contract can put an NBA franchise in a hole that can take up to a decade to recover from.

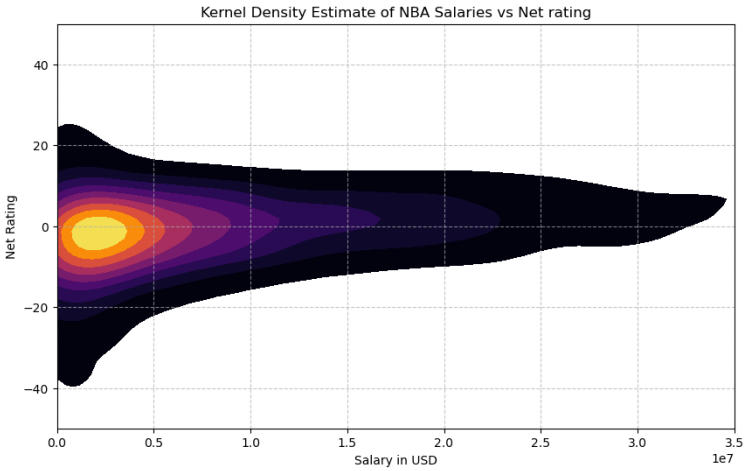

Figure 1: plot of NBA salaries vs net rating

While there is a relationship between net rating, a measurement of a team's performance when a player is on the court subtracted by team output when

they are off the court, and salary, it would be expected that this would be a much starker correlation. In Figure 1 there is a clear drop off in net rating as salaries near 250,00 USD per year, and there is a slight upward trend to 30,000,000 USD per year with a significant jump between 30,000,000 and 35,000,000 USD per year. Ideally, as the higher salaries charted on this plot are approached, especially after 20,000,000 USD per year, it would be expected that there will be very few players in these pay ranges who have a negative net rating, yet this is not the case.

Our proposal to rectify the described discrepancy is to create a model that takes various player statistics as an input and returns a predicted player output and an estimation of what a successful front office would compensate a player with this output.

III. Methodology

a. Data procurement

The first step in developing a solution to this complex issue was to select a dataset that would suit our needs. In this case we had to combine 3 datasets from different sources to create a dataset with the required features.

The first dataset was from Kaggle and contained all individual player statistics for each player year combination between 1996 and 2021 (for example: Dennis Rodman 1996/97 is treated as a different unrelated observation as Dennis Rodman 1997/98). The second dataset was a dataframe of NBA player salaries and NBA player salaries adjusted for inflation. This was procured from Spotrac, a website that specializes in contract information for major sports leagues in the US and worldwide and the final dataset was comprised of team statistics by year from basketball reference, a leading compiler and publisher of basketball statistics.

The first step in cleaning this data was to perform an SQL join on the first and second dataframes on ‘Player_ID’ and ‘Year’, this was followed by an additional SQL join between the newly formed dataset and the third dataset on ‘Team_abbreviation’ and ‘Player_ID’. This provided a dataframe with all the information necessary to begin IDA/EDA

b. EDA/IDA

First the output prospective output features were visualized, identifying any skewed or outlier rich features would guide our model selection and hyper parameter tuning going forward, this is represented in Figure 2 (below)

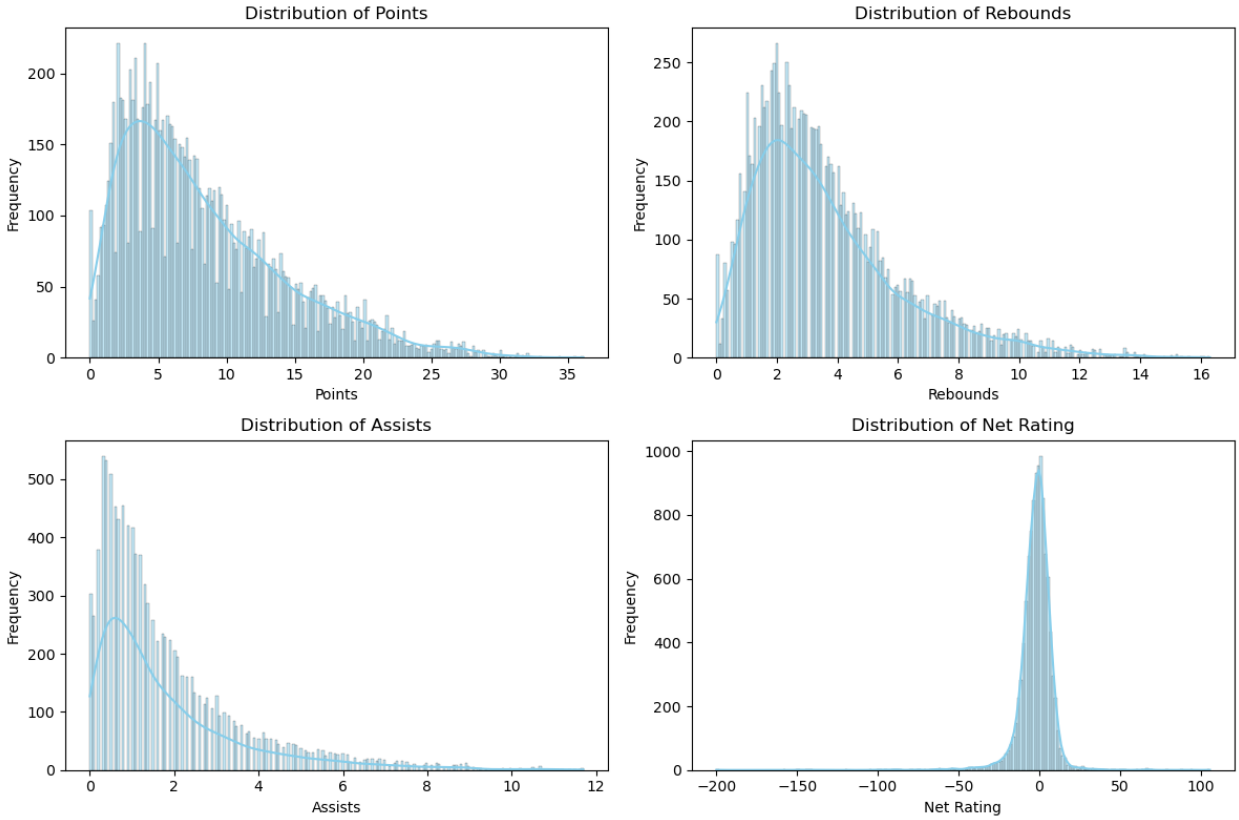

Figure 2: 4 perspective output features

'Points' and 'rebounds', both natural choices as comparative outputs for a basketball player, though skewed, both followed a normal distribution. Whereas 'assists' was closer to resembling an exponential distribution, and 'net rating' contained a significant number of extreme outliers.

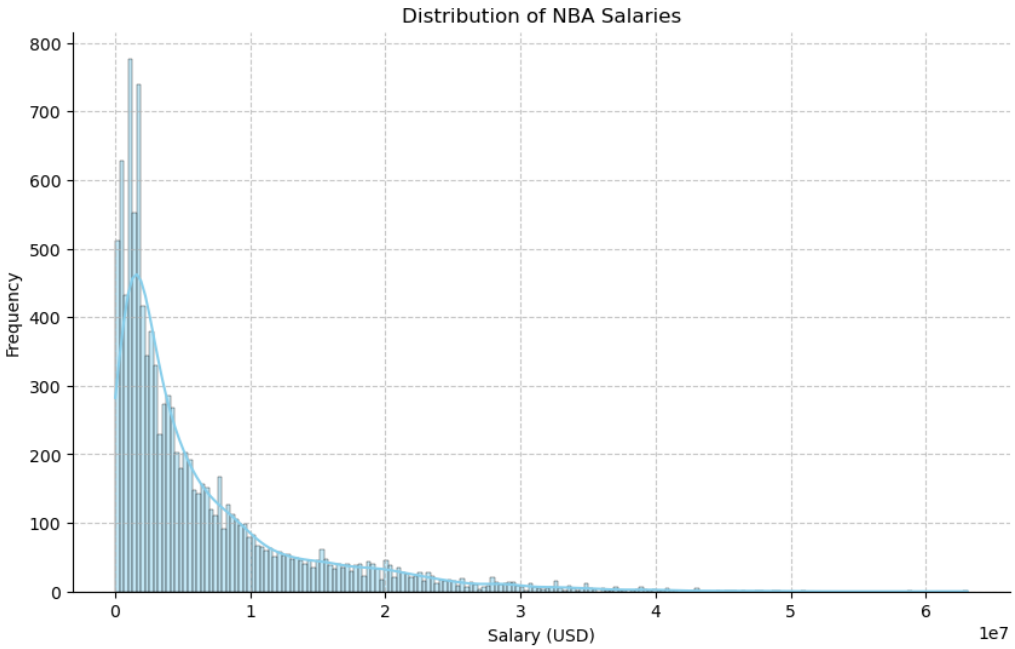

Figure 3: Plot of NBA player salary (adjusted for inflation)

Another key output feature for our model was salary (see Figure 3, above). The 'salary' feature was very outlier heavy and necessitated cross validation to be integrated into any future models. Another example of this can be viewed in Figure 4 (below), which displays a breakdown of distribution of total salary for the top 10% of players vs the field.

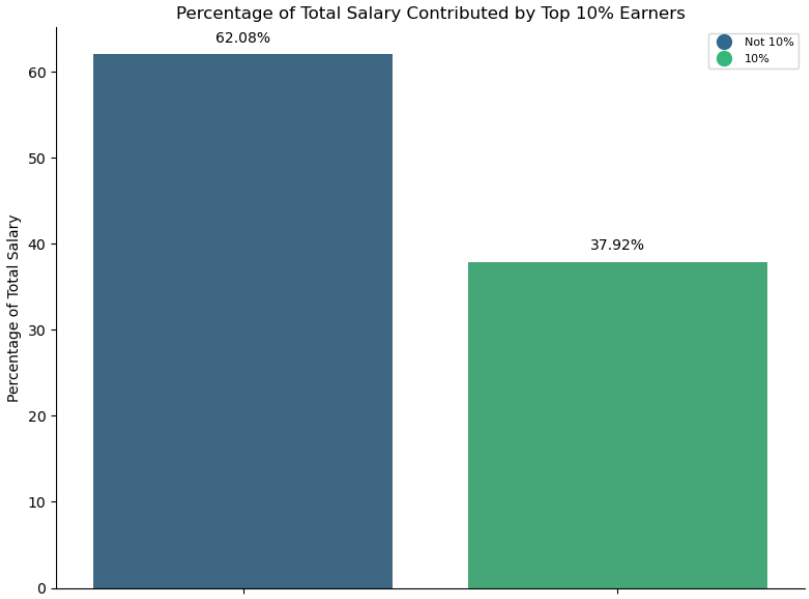

Figure 4: Salary breakdown of NBA players

After some deliberation we decided to proceed with a 2-stage model, as predicting the output of a player and the appropriate salary was a complex task that required a multifaceted approach that was likely out of the scope of a single model.

c. Model 1 – Determining player output

The first model would take a variety of features as an input and return a prediction of on court performance (in the form of hard statistics such as points, rebounds, assists, etc)

i. *Data curation*

The first step in building the dataframe was creating appropriate input features for our application. This first consisted of one-hot-encoding of categorical variables that were strings and creating lag data. We decided to lag the output features for 1 year and 2 years in the past for each player. This process created a dataframe that contained 4 distinct categories of features: unlagged static player features (Age, height, weight, country, etc), target features (points, rebounds, assists, etc), 1 year lag of target features, 2 year lag of target features. Following this several irrelevant features were dropped along with all string features that were not encoded. The final features are represented in a correlation heatmap. (Figure 5, page 6). Following the preparation of the dataframe our data was split into train and test subsets at a ratio of 80/20 with our target variables being separated from our input variables.

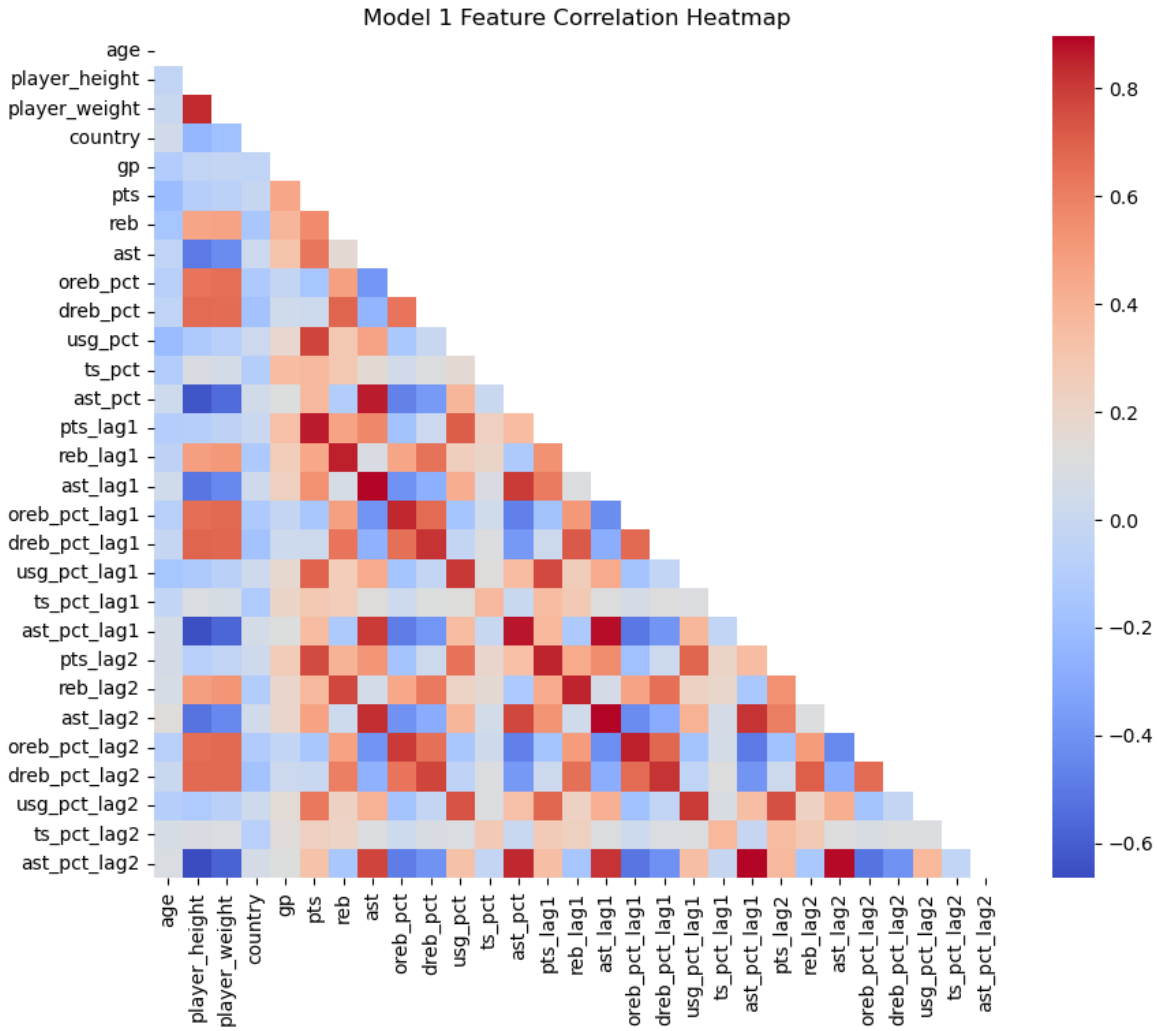

Figure 5: Correlation heatmap of data prepared for Model 1

ii. *Base model selection*

To determine which model we would spend time with hyperparameter tuning and other improvements, we created a pipeline of the following models with default parameters:

- Linear regression
- Random forest regression
- XGBoost
- SVC

These models were selected due to their ability to generate continuous as it did not make sense to utilize a model that excels in classification for our use case. A 5-fold cross validation was utilized for each of these models to

help deal with outliers in our features, in addition to ensuring that our models were not overfitting.

Of these models the Random Forest Regressor outperformed the other initial models, with the linear regression model performing the worst. More on how this was measured in the 'Results' section.

iii. *Hyperparameter tuning*

RandomSearchCV was our choice of hyperparameter tuning packages, as GrisSearchCV was computationally very expensive in comparison.

Figure 6 (below) displays the parameter grid utilized for the RandomSearch, while Figure 7 (below) displays the output parameters from the random search. After applying these parameters we saw an increase of less than an average of 1% across our output results.

```
param_dist = {
    'estimator__n_estimators': randint(50, 100, 200),
    'estimator__max_depth': [None, 10, 20],
    'estimator__min_samples_split': randint(2, 7),
    'estimator__min_samples_leaf': randint(1, 2)
}
```

Figure 6: RandomSearchCV parameters

```
MultiOutputRegressor(estimator=RandomForestRegressor(max_depth=20,
                                                     min_samples_split=4,
                                                     n_estimators=80,
                                                     random_state=42))
```

Figure 7: output from RandomSearchCV

iv. *Autoregressive iteration*

The final step for our first model was autoregressive iteration. Currently, we have a model that predicts the output of a player based on a combination of lagged performance data and general information. Without our autoregressive iteration step the goal is to be able to predict more than 1 year in the future. This involved setting up a for loop with a year variable. Then for each year in the year variable the loop would do the following:

- Remove the lag_2 data
- Push lag_1 data to lag_2

- Push current predictions to lag_1
- Repredict current predictions

The inputs and outputs are shown in Figures 8 and 9 (below)

```
new_inputs = {
    'age': 28.7,
    'player_height': 201.05,
    'player_weight': 101.4,
    'country': 1,
    'draft_round': 1,
    'draft_number': 10,
    'gp': 56,
    'pts_lag1': 9.9,
    'reb_lag1': 4.2,
    'ast_lag1': 2.1,
    'oreb_pct_lag1': 0.055,
    'dreb_pct_lag1': 0.14,
    'usg_pct_lag1': .13,
    'ts_pct_lag1': 0.52,
    'ast_pct_lag1': 0.14,
    'pts_lag2': 9,
    'reb_lag2': 4,
    'ast_lag2': 2.1,
    'oreb_pct_lag2': 0.08,
    'dreb_pct_lag2': 0.12,
    'usg_pct_lag2': 22,
    'ts_pct_lag2': 0.58,
    'ast_pct_lag2': 0.13
}
```

Figure 8: Dictionary of potential player inputs

	Pts	Reb	Ast	Oreb_pct	Dreb_pct	Usg_pct	Ts_pct	Ast_pct
0	8.716	4.255	1.710	0.05517	0.13605	0.15218	0.52545	0.12323
1	7.569	3.966	1.455	0.05055	0.13671	0.15147	0.51433	0.10719

Figure 9: dataframe of Model 1 output for year 1 (index 0) and autoregressively iterated year 2 (index 1)

d. Model 2 – Player salary prediction

i. Data curation

Model 2 required different inputs and outputs than the original dataframe, as such the features that were selected were different. From the original dataset all observations that had team winning percentages < .66 were omitted. We are only interested in viewing contracts in the context of

successful teams. All features that were strings were either encoded or removed from the dataset. Additionally, all features that were not either the inputs (which are the same format as the outputs from model 1) or salary (the output of model 2) were omitted from the final dataframe. The data for model 2, similarly to model 1, was split into train and test subsets at a ratio of 80/20 with salary being segregated as the output and the output from model one being the input variables.

ii. *Base model selection*

The same pipeline employed for the creation of the model was utilized for the creation of Model 2. In this case the Linear Regression model had superior output when compared to the other models in the pipeline. Cross validation was once again utilized as salary was a feature heavy in outliers. Hyperparameter tuning was attempted for Model 2 in the form of ridge regression, but this yielded no usable results, as such the base model was selected as the final Model 2.

IV. Results

Success was measured differently for each of the models. Model 1 had output variables that were skewed normal distributions whereas the output variable for Model 2 had an extremely outlier heavy output variable. This made RMSE and MSE a reliable measurement of success but applying these statistical measurements to Model 2 could cause issues.

Performance Metrics Model 1:

Output	MSE	RMSE
Pts	7.1563	2.6751
Reb	1.4598	1.2082
Ast	0.5891	0.7675
Oreb_pct	0.0004	0.0196
Dreb_pct	0.0011	0.0327
Usg_pct	0.0008	0.0276
Ts_pct	0.0045	0.0673
Ast_pct	0.0020	0.0452
Average	1.1517	0.6054

Figure 10: MSE and RMSE values for output features of Model 1 (test data)

With domain knowledge of NBA player statistics, it is evident that these are acceptable errors for the type of values Model 1predicts. Normalized RMSE values also portray this, for example the range of the features 'points' is 38.1 for this dataset. This means that the RMSE for points in our model (2.675) only covers ~7% of the total range of the variable, which is an acceptable variation for the application.

Salary (in USD, adjusted for inflation) was the singular output for Model. Due to the significant number of outliers in the salary feature we opted to use Median Average Error as the measurement of success, as it was more forgiving to these conditions.

```
Model 2: median average error: 2662071 USD
```

Figure 11: Model 2 median average error

On a normalized basis the MAE only covers ~ 6% of the range of NBA salaries, paired with the knowledge that 2.6m USD dollars is less than 2% of total NBA salary cap, this makes the error for this model reasonable for the application as well.

V. Lessons learned

Several important lessons were learned during the development of this project. During the model selection portion, a major overfitting issue was identified with several of the neural network models that were initially considered. For brevity the process of discovery was not included under methodology. These overfitting issues were discovered by testing the train set on the model, with the returned outfit having a MSE of 0.0 for more than half of the output variables.

There were also some important learning moments surrounding hyperparameter tuning and computational taxation. GridSearchCV proved to be too much for our laptops to handle, this could be rectified in the future by using HPCC/ICER. We also found that linear regression hyperparameter tuning can prove difficult if you have little experienced with ridge and lasso

regression, our model was not improved by using these techniques and I believe that comes down to potential user error. When it comes to hyper parameter tuning it can take a significant knowledge base of specific models and exactly how they function to develop parameter grids that will improve models.

VI. Future improvements

By far the biggest future improvement for this project would be to find a server to deploy this as an application available for anyone to use, from amateur fans to professional front offices. This would involve integration of the autoregression feature directly into the second model. Ideally this could deployed in a format where the user could input player information and and a desired number of years and the output would be a fully fleshed out NBA contract.

The other part of this future project could be a paraellel model that performs the opposite function, i.e, input a contract and a classification model would tell the user if this was a wise contact to offer or not.