Task A Report - Moneyball

SID: 490411065

1 Executive Summary

Machine Learning (ML) techniques are being used to solve business problems in a data-driven way in every industry you can think of. The National Basketball Association (NBA) is certainly no exception — obtaining the best possible basketball team is a highly lucrative problem that can be solved in a data-driven way.

Specifically, NBA teams are always looking for novel ways to reduce their salary expenses, because they must pay players in accordance with the NBA salary cap. Optimisation problems like these are best solved using ML techniques as there are plenty of factors that influence such costs.

In this report, we provide three ML methods that may be used to predict salaries of NBA players using their basketball stats. We begin by running through an initial analysis of an NBA data set to look for patterns between NBA salaries and basketball stats. We then use these patterns to inform our 'feature engineering' decisions and model selection procedures. Finally, we evaluate our chosen models on a fresh, new test data set and analyse the results.

As a quick summary, we employ k-Nearest Neighbours (kNN), linear regression, and lasso algorithms to select three best-performing models. We have been informed that two of our models must have a maximum root-mean-square error (RMSE) of 4.1 million dollars on the test data set. Fortunately, we are able to achieve this target, as our kNN and lasso models attain RMSEs of ~4.06 million dollars, and our linear regression model comes in really close at 4.1001 million dollars.

If only a single model must be used, we advise that the kNN model may have a slight edge over the lasso model because it makes fewer modelling assumptions.

2 Exploratory Data Analysis

2.1 The Training Set

Before we start playing around with our data, let's take a minute to understand the origins of our data and what variables it contains.

The training data set — the data set that we use in sections 2 *Exploratory Data Analysis* and 3 *Methodology and Modelling* — contains 126 examples and was sourced from the official NBA Advanced Stats website at https://stats.nba.com. Apart from the `Train_ID` column, the data set contains the following variables (all based on a single NBA season):

- `SALARY` (numeric): Dollars earned in millions
- `POSITION` (categorical): Position played on the court
- `TEAM` (categorical): Team played for
- `Age` (numeric): Age in years old
- `Games` (numeric): Games played
- `Minutes` (numeric): Total minutes played
- `PER` (numeric): Personal efficiency rating
- `TS` (numeric): True shooting percentage
- `ORB` (numeric): Offensive rebounds
- `DRB` (numeric): Defensive rebounds
- `TRB` (numeric): Total rebounds
- `AST` (numeric): Assists
- `STL` (numeric): Steals
- `BLK` (numeric): Blocks

- `TOV` (numeric): Turnover percentage
- `USG` (numeric): Usage percentage
- `ORtg` (numeric): Offensive rating
- `DRtg` (numeric): Defensive rating
- `OWS` (numeric): Offensive win shares
- `DWS` (numeric): Defensive win shares
- `WS` (numeric): Win shares

Since the purpose of this report is to use ML algorithms to predict salary, `SALARY` is the response and all other variables are predictors.

2.2 Data Processing

We need to make sure that our data set is free of errors before we perform any analysis. Since the data is sourced from the official NBA site, this should be the case but we still need to do our due diligence.

We start by checking how many (non-missing) values are in each variable:

	SALARY	POSITION	TEAM	Age	Games	Minutes	PER	TS	ORB	DRB
count	126	126	126	126	126	126	126	126	126	126

	TRB	AST	STL	BLK	TOV	USG	ORtg	DRtg	OWS	DWS	WS
count	126	126	126	126	126	126	126	126	126	126	126

Figure 2.2.1

Each variable has 126 values — the same as the number of examples in our data set — so we can safely assume that there are no missing values in the data.

Next, we plot histograms of the numeric variables to see if there are any extreme values. Below are histograms for `SALARY`, `Age` and `Minutes`:

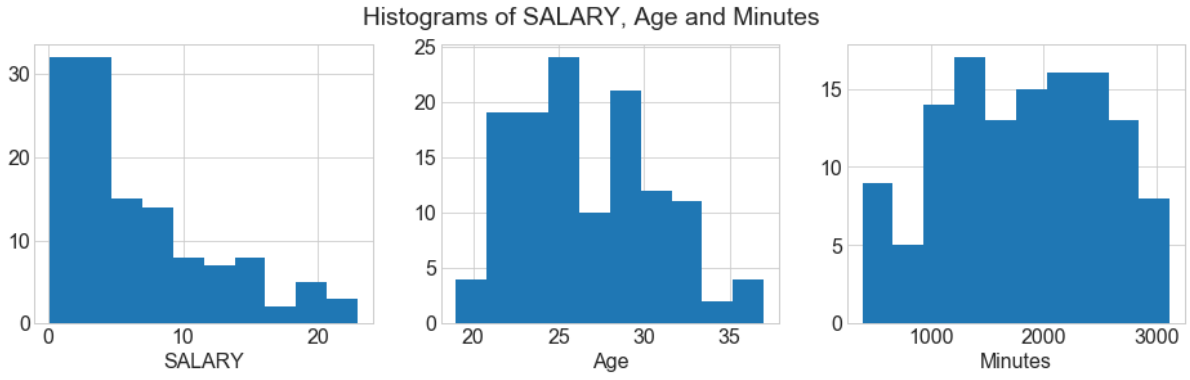

Figure 2.2.2

Note that all of the numeric variables in our data set measure some kind of human characteristic or performance metric, so we should expect them to follow a normal or skew-normal distribution. This appears to be the case in the above histograms, and we especially note the absence of any extreme values. We leave the rest of the histograms in the appendix (Figure A1) as it is easy to check that none of the other numeric variables contain any extreme values.

Finally, we plot the label frequencies of the two categorical variables `POSITION` and `TEAM` to check for any invalid or mis-typed labels:

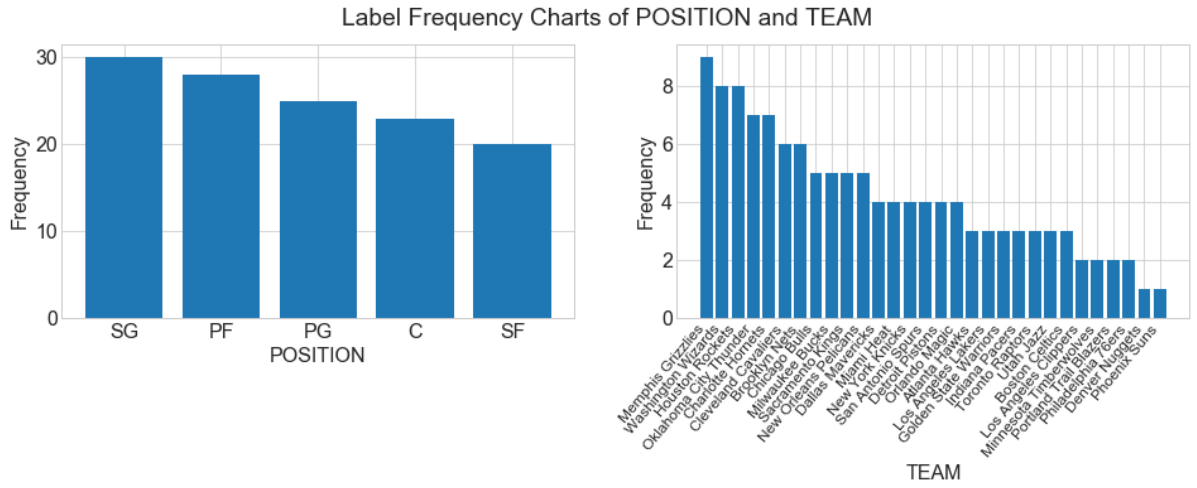

Figure 2.2.3

SG (Shooting Guard), PF (Power Forward), PG (Point Guard), C (Centre) and SF (Small Forward) are all valid basketball positions. It can also be verified that all of the team names are valid NBA teams using this teams list: https://www.nbateamslist.com/.

And with that, we complete our due diligence. We safely assume that there are no missing, extreme or invalid values that could have been caused by typos in the data set.

2.3 Simple Analysis

In this section, we try to find patterns and relationships in our data that we can exploit in section 3 *Methodology and Modelling* to improve our ML algorithms.

We start by looking at the (linear) correlation coefficients between each of the numeric variables:

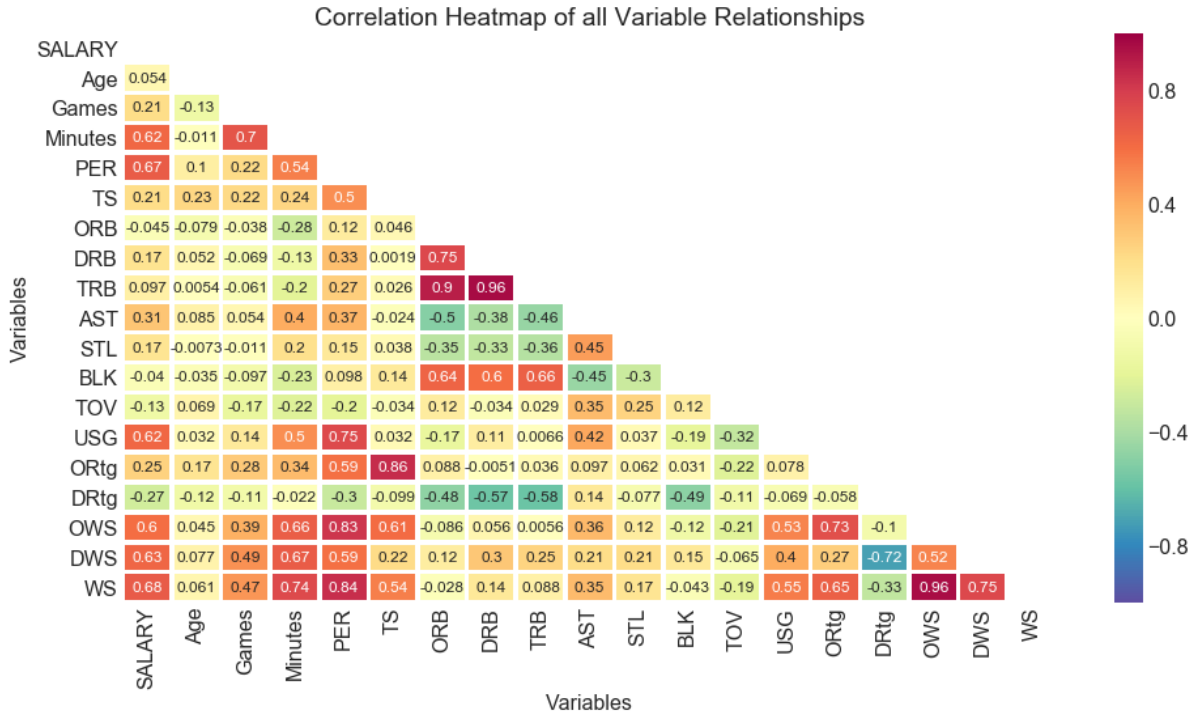

Figure 2.3.1

A few variables have more than 0.6 correlation with the response SALARY but most variables have low correlation with SALARY. We also see that some variables are very highly correlated with each other, namely

0.96 between `WS` and `OWS`. This suggests potential multicollinearity issues, implying that not all variables are independent of each other. We need to look into this.

Variance Inflation Factors (VIFs) offer a good way of diagnosing multicollinearity problems. They are calculated using the R^2 value that comes from regressing a predictor on all other predictors. As a general rule of thumb, a VIF greater than 10 indicates severe multicollinearity problems. Figure A2 in the appendix shows that many predictors have extremely high VIFs (in the hundreds and thousands), suggesting high linear dependence between them. However, as we iteratively remove predictors with high VIFs and recalculate the VIFs, the VIFs of the other predictors decrease. Below are the VIFs of the predictors that remain after removing some problematic predictors:

Predictors	VIF
Age	1.3079
Games	2.8309
Minutes	4.6010
TS	2.8793
ORB	3.5614
DRB	3.7416
AST	4.1792
STL	1.6801
BLK	2.3820
TOV	2.5019
USG	2.4886
DRtg	2.0793
OWS	4.9438

Figure 2.3.2

These VIFs are all under 5, which means that there are no severe multicollinearity problems in this set of predictors. This does not mean that we have solved all dependence problems, because all of our predictors to some degree depend on the latent/unmeasurable variable *basketball skill*. We will just have to keep this in mind when interpreting models later on.

Note that we are not removing any variables from our data set, we are just looking for patterns between combinations of variables. Next, we investigate scatter plots between numeric predictors and `SALARY`. Below are scatter plots involving `WS`, `TRB` and `Minutes`:

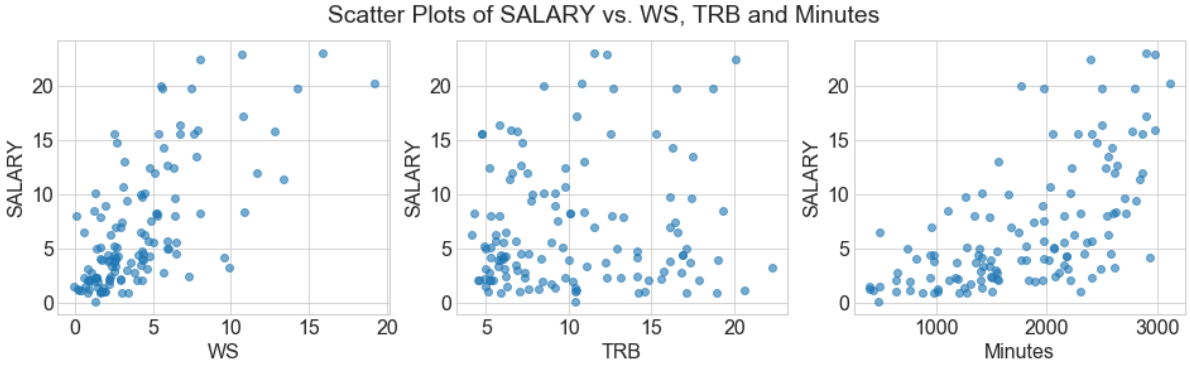

Figure 2.3.3

Although there may be underlying relationships with `SALARY`, there is too much noise to see any strong patterns in the scatter plots. At best, there is a vague linear pattern (e.g. `WS`) and at worst, there is no pattern (e.g. `TRB`). `Minutes` hints at a possible quadratic relationship but again, there is just too much noise. Plots of the other predictors vs. `SALARY` show a similar lack of noticeable pattern, and are provided in Figure A3 in the appendix.

Side note: taking the logarithm of `SALARY` before producing scatter plots does not help much with discerning any patterns. These plots are also provided in the appendix (Figure A4).

Finally, we look at how well our categorical predictors relate with `SALARY`. We construct a box plot of `SALARY` for each label of the `TEAM` variable:

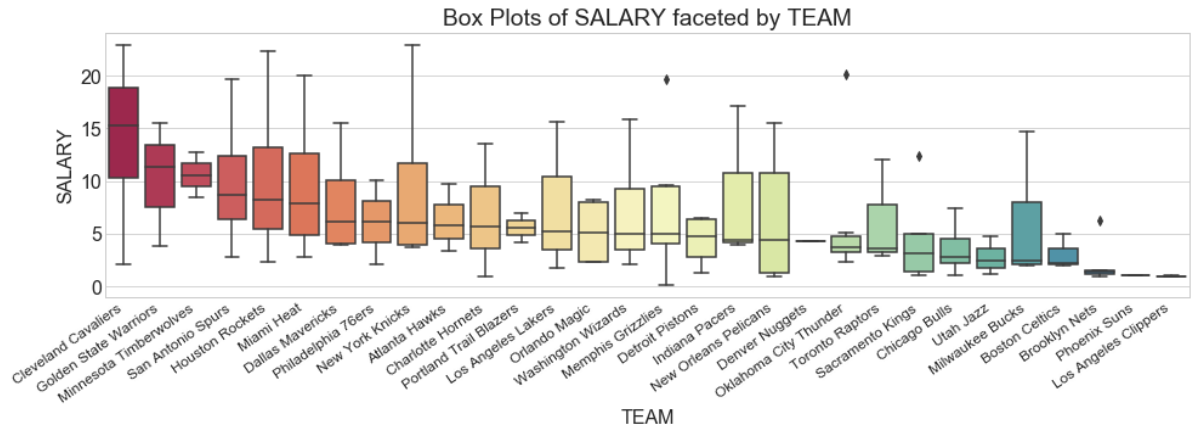

Figure 2.3.4

The ranges of the boxes are all over the place, but the median salary of Cleveland Cavaliers players seem a bit higher than other teams' players. We have left the box plots for the `POSITION` variable in the appendix (Figure A5) since the median salaries for the positions are almost the same.

We now conclude our EDA section. Depending on the ML algorithm and its built-in assumptions, we will refer back to this section to justify any choices we make when selecting variables and refining algorithms in the next section.

3 Methodology and Modelling

3.1 Preliminary Feature Engineering

The purpose of feature engineering is to provide a bridge between EDA and model fitting by transforming data features (i.e. predictors) so that they

1. are able to be processed by ML algorithms (e.g. dummy-encoding categorical predictors),
2. better capture relationships seen in EDA plots (e.g. squaring a predictor), and
3. satisfy model assumptions (e.g. by standardising predictors).

For our preliminary feature engineering, we only address the first point. We then perform additional feature engineering tailored to each of our three ML algorithms to address the other two points, in later sections.

The two categorical predictors in our data set are both un-ordered, so we use a binary encoding system to transform the categorical predictors into sets of dummy variables. For example, we transform `POSITION`

into a set of dummy variables with the following label encodings (note that we avoid the 'dummy variable trap'):

	D1	D2	D3	D4
PG	1	0	0	0
PF	0	1	0	0
SG	0	0	1	0
SF	0	0	0	1
C	0	0	0	0

Figure 3.1.1

We do a similar thing with the `TEAM` variable. Now all predictors are of a numerical form and thus can be processed by ML algorithms, so we are finished with our preliminary feature engineering.

3.2 k-Nearest Neighbours

3.2.1 Additional Feature Engineering

As mentioned in section 3.1, we will do some extra feature engineering on our data before we utilise the k-Nearest Neighbours (kNN) algorithm.

The kNN algorithm needs to calculate distances, so it implicitly assumes that all predictors are roughly on the same scale — Since our data's values vary widely from predictor to predictor, we need to standardise all of our predictors.

Next, we narrow down our predictor set. We want to reduce any adverse effects that may arise from the 'curse of dimensionality'. kNN works best with predictors that have a *continuous curve*-like relationship with `SALARY`. However, as we saw in the scatter plots of section 2.3, there were no significant non-linear patterns. Hence, we only keep the predictors that had the strongest linear relationship with the response variable, i.e. the ones with the highest linear correlation. These predictors are `Minutes`, `PER`, `USG`, `DWS` and `WS` — all of which have a linear correlation greater than 0.6 with `SALARY`.

3.2.2 Model Selection

We now arrive at the modelling section where we explain our model selection process and showcase our best kNN model.

We first consider all of our five chosen predictors. We use the `GridSearchCV` class from scikit-learn to find the model with the value of k — the number of neighbours — that minimises its 5-fold cross-validation (CV) root-mean-squared error (RMSE). Then, we go through these steps again with all possible subsets of our five predictors, in case there is a smaller combination of predictors that perform better than the original five.

We iterate over 100 values of k for each of the $2^5 - 1 = 31$ variable combinations, resulting in $100 \times 31 = 3100$ models validated. The model with the lowest CV RMSE score is re-trained on the full training data set, and its model specification is shown below:

```
KNeighborsRegressor(
    X=train['Minutes', 'PER', 'USG', 'DWS'],
    y=train['SALARY'],
    n_neighbors=11
)
```

```
CV RMSE: 3.6161
```

Our final model uses $k = 11$. Since we used cross-validation to choose the optimal model, we can be confident that our model optimises the theoretical bias-variance trade-off curve. If we did not use proper validation techniques and instead had fit our 3100 models on the full training set, we would've ran a high risk of selecting an overfit model. This model would have the lowest training set bias but potentially high variance, and would not generalise well to unseen data.

3.2.3 Checking Assumptions

kNN is a non-parametric algorithm. All it requires is a measure of distance and that predictors are roughly on the same scale. Our selected model was trained using the default Euclidean distance metric, and our predictors were standardised prior to training the model. Hence, we have satisfied the main kNN assumptions.

3.3 Linear Regression

3.3.1 Additional Feature Engineering

With linear regression, you are generally looking to capture non-linear relationships that exist between each predictor and the response. However, as we saw in the scatter plots of section 2.3, there were no non-linear patterns that stood out to us. We therefore do not transform any predictors.

We do however, want to deal with multicollinearity problems beforehand. In our EDA, we came up with a set of predictors that all had VIFs lower than 5. We choose to only keep these predictors as we move onto the model selection stage.

3.3.2 Model Selection

Amongst our chosen predictors, we will search for a good subset of predictors using *forward selection*. The first part of the forward selection procedure is as follows:

1. Fit p linear regression models using one of your p predictors at a time
2. Take note of the predictor that gives the lowest training set RMSE
3. Fit $p - 1$ linear regression models using the noted predictor and one of your remaining $p - 1$ predictors at a time
4. Take note of the two predictors (one new, one from step 2.) that give the lowest training set RMSE
5. Continue noting down predictor combinations in a similar manner until there are none remaining

We have 13 predictors, which equates to $\frac{13 \times 14}{2} = 91$ models fit on the training set. The above steps narrow these 91 models down to 13 candidate models, i.e. the best model with 1 predictor, the best model with 2 predictors, etc. We then go through the last part of the forward selection procedure:

1. Perform cross-validation to calculate a CV RMSE for each of the p candidate models
2. Select the model with the lowest CV RMSE

These last validation steps are important, because they allow us to select the model with the optimal complexity, i.e. the optimal number of predictors. Below is a plot of the number of predictors in our model versus its training set RMSE and CV RMSE:

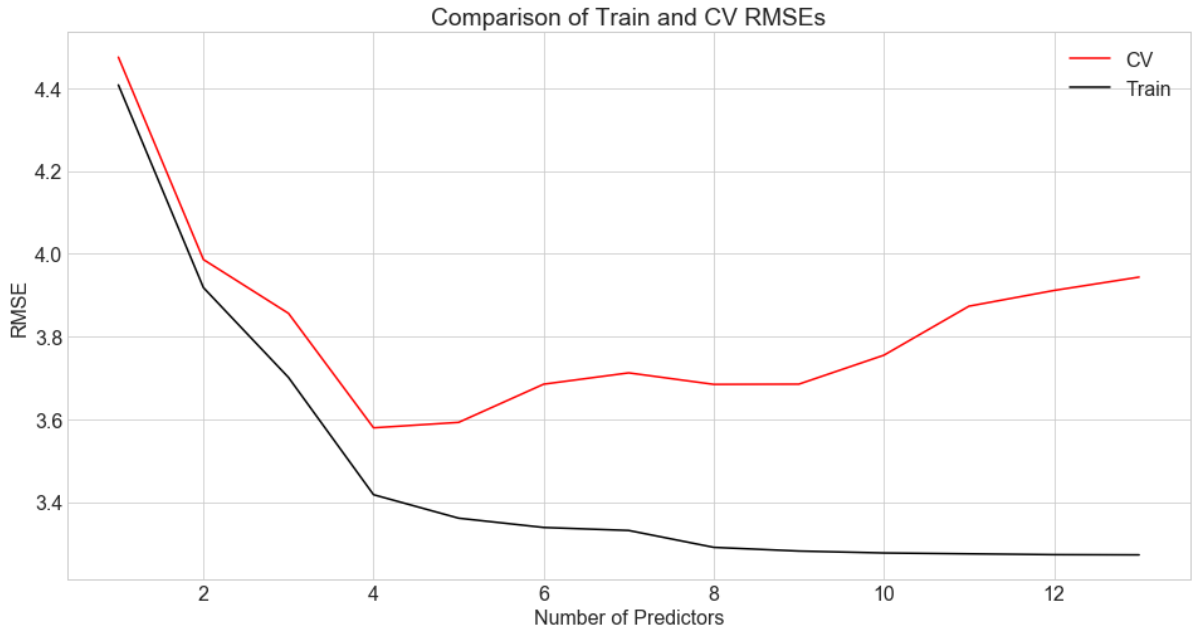

Figure 3.3.2.1

As we can see, the cross-validation steps prevent us from picking a model that is too complex and potentially overfit. Our final model uses 4 predictors and is specified below:

```
LinearRegression(
    X=train['Minutes', 'USG', 'DRtg', 'Games'],
    y=train['SALARY]
)

CV RMSE: 3.5792
```

3.3.3 Checking Assumptions

Linear regression models are a bit tricky when it comes to assumption checking. There are plenty of statistical assumptions regarding normality, exogeneity, finite 4th moments, etc. However, we are not operating in a statistical context; we want to be as practical as possible with our model building and therefore only assume that the relationship between `SALARY` and the predictors can be modelled with a hyperplane. We did not find any strong evidence against this in our EDA.

Multicollinearity between predictors inflates the variability of our model predictions, so we made sure to control for this in the feature engineering section by choosing low-VIF predictors.

We also used proper variable selection techniques and accounted for overfitting issues by cross-validating models with different complexities.

3.4 Lasso

3.4.1 What is Lasso?

In section 3.3.2, we used forward selection to select the best linear regression model. More specifically, forward selection provided a way of optimising our model's *complexity* by choosing the right amount of predictors.

The lasso algorithm is quite similar to linear regression, in the sense that it also fits a hyperplane to data. However, it has an in-built way of optimising model complexity that involves *regularisation* instead of 'choosing the right amount of predictors'.

Regularisation in lasso (more specifically, L^1 regularisation) is achieved by minimising a 'complexity' objective; a function of the β coefficients:

$$\alpha\left(|\beta_1| + |\beta_2| + \cdots + |\beta_p|\right)$$

This complexity objective is in addition to the least-squares objective that linear regression tries to minimise. In summary, lasso is linear regression with built-in complexity optimisation, in the form of regularisation.

3.4.2 Additional Feature Engineering

The complexity objective in lasso assumes that the β parameters are on a relatively similar scale. This is similar to what we needed back in the kNN algorithm, so we have to standardise all of our predictors.

We did not transform any predictors in the linear regression section, so we do not do that here either.

Since lasso already optimises model complexity, we do not need to omit any predictors. The meaningful predictors will be given higher β coefficient values.

3.4.3 Model Selection

Lasso models have a tuneable hyperparameter that controls the level of regularisation. This is the α constant that we saw in the complexity objective of section 3.4.1. The greater the value of α, the greater the level of regularisation, which in turn means a less complex model.

Since our response variable takes on relatively small values and we standardised our predictors, we expect α to be roughly between 0 and 1. Thus, we try 20 different values for α that are between 0 and 1 to select our model. We use the `GridSearchCV` class from scikit-learn to find the lasso model with the value of α that minimises its 5-fold CV RMSE. A plot of α versus the CV RMSE is shown below:

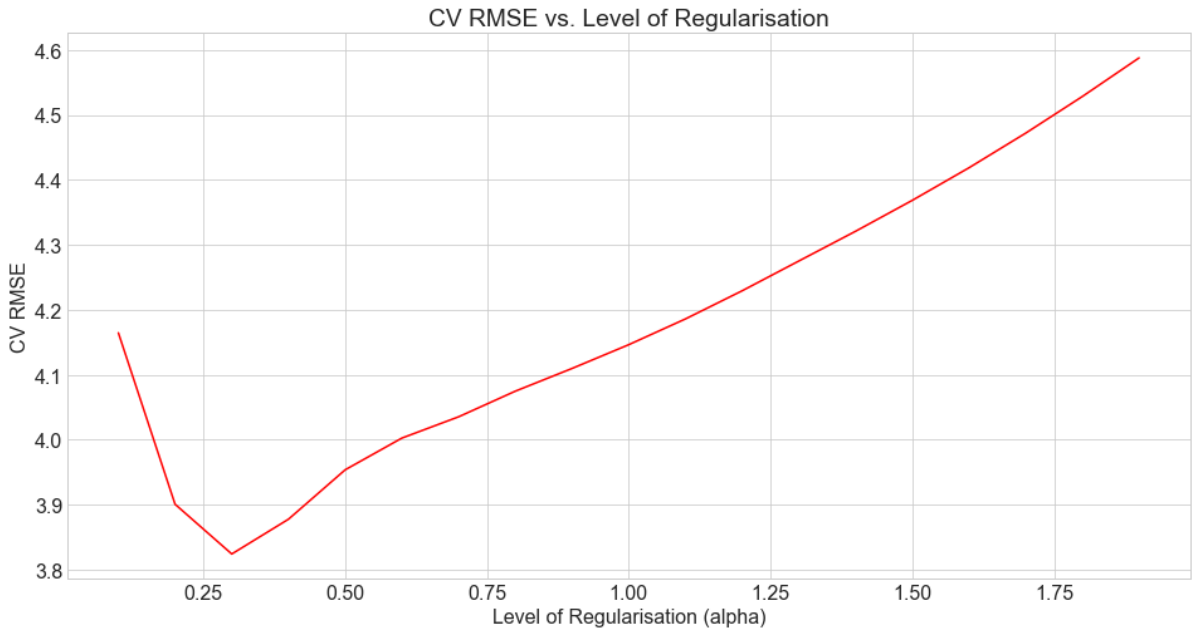

Figure 3.4.3.1

$\alpha = 0.3$ minimises the CV RMSE. Hence, our best lasso model specification is:

```
Lasso(
    X=train.loc[:, 'Age':],
    y=train['SALARY'],
    alpha=0.3
)
```

```
CV RMSE: 3.8243
```

Note that all predictors are used to train the model.

3.4.4 Checking Assumptions

As we know, lasso models are a special type of linear regression, so naturally we assume that the relationship between the response and the predictors can be modelled with a hyperplane. Our relatively low CV RMSE results show that this is a viable conclusion to make, even if this relationship is not directly causal.

Our selected model has a regularisation level of 0.3, so it successfully mitigates multicollinearity issues by shrinking the β coefficients of problematic predictors.

4 Results

4.1 The Test Set

The test set has 127 examples and the same variables as the training set.

So far, we have only used the training set. We have performed EDA, model fitting, model validation, and selected three models all using the training set. We have not touched the test set. This was all to ensure that the scores that our selected models attain in the next section are all legitimate, since we will test these models on completely fresh, unseen data.

4.2 Model Evaluation

With our three selected models, we finally arrive at the model evaluation stage. We use the test set's predictors to predict SALARY values, and then calculate the RMSE with the test set's actual SALARY values. Our results, along with the previous CV RMSEs that we obtained earlier on the training set, are shown in the table below:

	CV RMSE	**Test RMSE**
kNN	3.6161	4.0645
Linear	3.5792	4.1001
Lasso	3.8243	4.0659

Figure 4.2.1

The first interesting thing to note here is that all of the test RMSEs were higher than the CV RMSEs. Since we selected our models using rigorous cross-validation techniques, we should expect the CV and test RMSEs to be quite similar. A possible reason for this not happening is that our test set was quite large. Usually in practice, the test set is around 10-20% of the size of the total data available, but in our case it was 50%. This may have resulted in too many unfamiliar scenarios for our selected models, meaning slightly worse predictions and higher test RMSEs.

Another thing to note is that the gap in CV and test RMSEs was the largest for the linear regression model, and was the smallest for the lasso model. It looks like our lasso model generalises better to out-of-sample data due to its regularisation capability.

Finally, we see that the kNN and lasso models both achieved test RMSEs under the target of 4.1, with linear regression being incredibly close. Choosing a best model is difficult here, and the choice really depends on how confident we are that a hyperplane should be used to model the relationship between SALARY and the predictors. While there wasn't strong evidence against this in our EDA, there wasn't strong evidence *for* it either. Hence, we are probably better off choosing a non-parametric model that does not make such assumptions, i.e. the kNN model.

Note that we are not trying to select a model in this section, we are just discussing the results of our chosen 3 ML models.

5 Conclusion

As we saw in section 4, we were able to meet the maximum test set RMSE that was assigned to us of 4.1 with our kNN and lasso models. We made sure to use 5-fold cross-validation in all of our model selection procedures to avoid overfitting problems.

Our selected models' CV RMSEs were all lower than their test RMSEs, and we believe this was only due to the 50-50 train/test split of our data set. A more traditional 80-20 split could have allowed for closer CV and train RMSEs.

With the two linear models (linear regression and lasso), we made sure that we accounted for multicollinearity issues by using forward selection and regularisation techniques. This was important because this reduced the complexity and statistical variability of our selected models. We similarly reduced complexity and statistical variability in our kNN model by using CV to choose k and by choosing only a few predictors that had high linear correlations with SALARY.

In conclusion, our three selected models all performed very well because we implemented a rigorous model selection process that involved cross-validating hundreds of models. With a larger training data set, more computing power and more specialised ML knowledge, we could produce even better models. Future work could involve trying more advanced methods such as neural networks, but we would first need to collect more data.

Appendix

References

- Cross Validated. (n.d.). k nearest neighbour - Why do you need to scale data in KNN. [online] Available at: https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn [Accessed 16 Oct. 2020].
- NBA Stats. (2019). Stat Glossary. [online] Available at: https://stats.nba.com/help/glossary/ [Accessed 14 Jan. 2020].
- NBA Teams List. (n.d.). NBA Teams List | Full List Eastern and Western Conference Teams. [online] Available at: https://www.nbateamslist.com/ [Accessed 16 Oct. 2020].
- Stack Overflow. (2019). Stack Overflow - Where Developers Learn, Share, & Build Careers. [online] Available at: https://stackoverflow.com/.
- Wikipedia Contributors (2019). Curse of dimensionality. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Curse_of_dimensionality.
- Anand, S. (2020). Dummy Variable Trap. [online] Medium. Available at: https://medium.com/datadriveninvestor/dummy-variable-trap-c6d4a387f10a [Accessed 16 Oct. 2020].
- Scikit-learn. (2019). scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation. [online] Available at: https://scikit-learn.org/stable/.
- pandas. (n.d.). pandas documentation — pandas 1.0.1 documentation. [online] Available at: https://pandas.pydata.org/docs/.
- numpy. (n.d.). NumPy Documentation. [online] Available at: https://numpy.org/doc/.
- Matplotlib. (2012). Matplotlib: Python plotting — Matplotlib 3.1.1 documentation. [online] Available at: https://matplotlib.org/.
- Seaborn Pydata. (2012). seaborn: statistical data visualization — seaborn 0.9.0 documentation. [online] Available at: https://seaborn.pydata.org/.

Figures

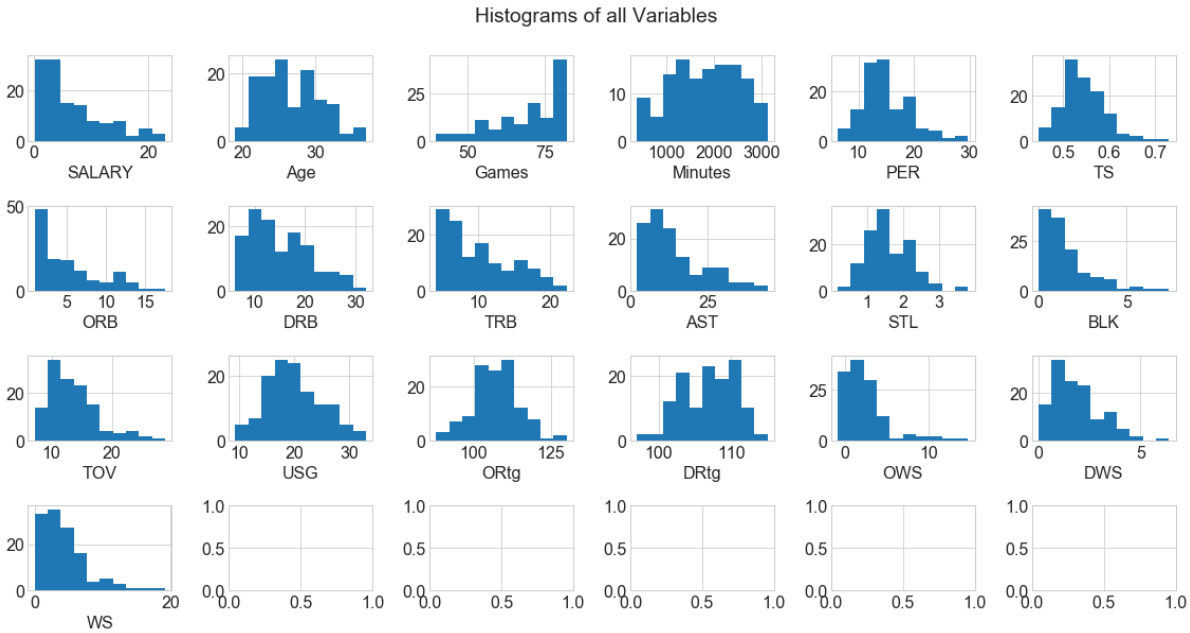

Figure A1

Predictors	VIF
Age	1.3668
Games	3.0245
Minutes	13.1751
PER	55.2824
TS	13.7395
ORB	317.1385
DRB	777.4770
TRB	1862.2961
AST	12.9470
STL	3.2680
BLK	3.5352
TOV	8.2269
USG	17.4443
ORtg	21.5102
DRtg	13.9834
OWS	2912.9543
DWS	560.9322
WS	4781.3676

Figure A2

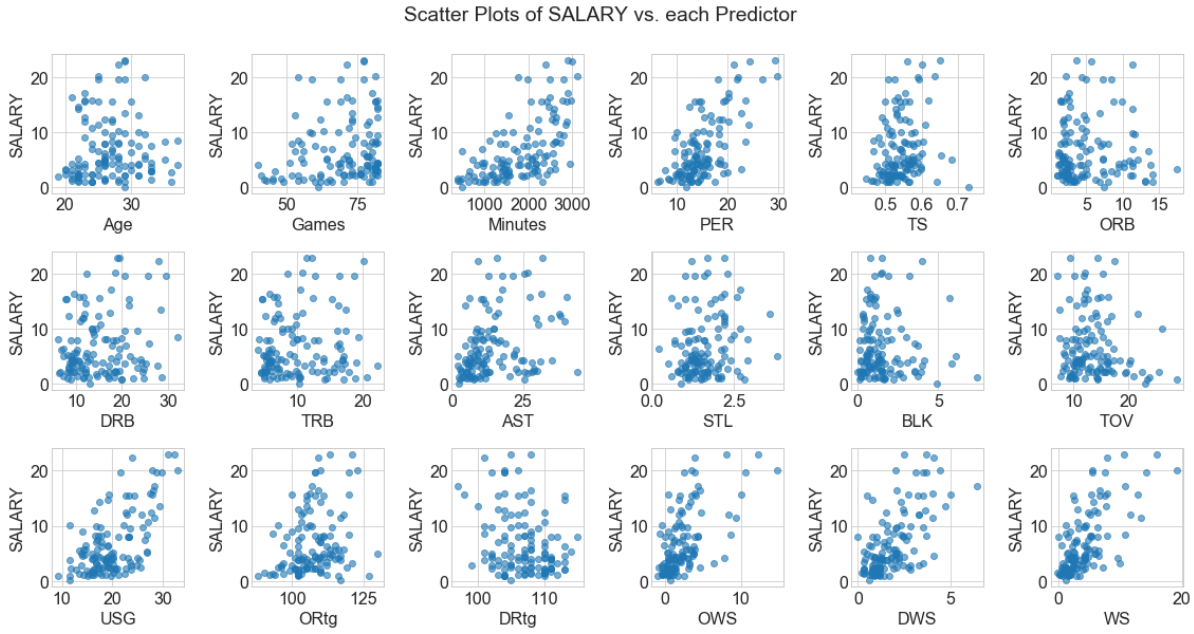

Figure A3

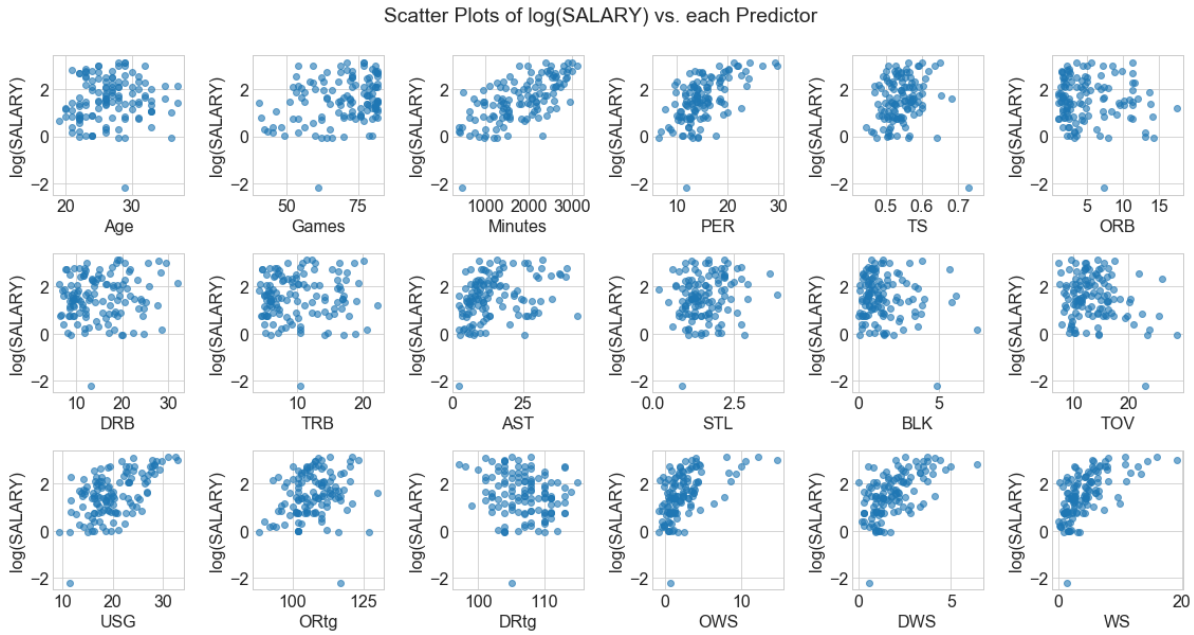

Figure A4


Box Plots of SALARY faceted by POSITION
SALARY
20
15
10
5
0
SF
SG
PF
C
PG
POSITION

Figure A5