COMP5310 Principles of Data Science
Project Stage 2

Student ID: 490196302, 490438970

Introduction

Problem:

The National Basketball Association is considered to be the top level basketball league in the world. Every year the league organises a draft lottery and 60 players are selected to join the league, within this 60 players around 50 players are selected from National College Athletic Association **(NCAA 2020)**. The player performance in the NCAA basketball league became a good indication on whether a player has a chance to enter the NBA. In basketball the most talked on court statistic is the points a player can score, thus it is quite important if we can know a players points production in game given some other on court statistic. In our study, we used machine learning algorithms to predict the points which a player can score in a game given some other on court performance. Furthermore, we are also interested in if a player has a chance to get picked into the NBA and if so, at what possible range. We have grouped pick into three classes: 'under 15', '15 to 30' and '30 to 60'. Therefore, classification models are used to classify a player's pick.

Workflow:

The dataset("CollegeBasketballPlayers2009-2021") is used for this study. Data cleaning will drop the rows with empty values and some personal information columns such as **"names"**. After data cleaning we got 1139 rows of data and 55 features related to a single player on court performance. Then the pre-processing techniques including normalisation, standardisation and principal component analysis are implemented on the dataset. In terms of model selection, we have chosen regression models to predict the player's average points production and classification models to predict the player's pick position.

Hypothesis:

For each research question, since we have chosen three different models, we want to compare if the models perform differently when predicting outputs. There is one benchmark model chosen for each of the questions whereas the benchmark model for regression problem is linear regression and for classification problem is k-nearest-neighbour. Therefore, the hypothesis test are set as below:
H0: The selected model and benchmark model have same performance in predicting output
H1: Two models have significantly different performance in predicting output

Evaluation:

In this study, the performance of the regression models will be assessed on the standard error. For classification models f1-score, accuracy score will be used. Furthermore, the reliability of the regression models will be qualified using r-square, f1-score will be used on classification models. We will obtain a list of 30 metrics formed by r-square in the regression model and the f1-score in the classification models with 30 fold cross validation. We will perform a pairwise t-test with 95% confidence interval to check on our hypothesis. If the p-value is less than 0.05, the null hypothesis is rejected, otherwise we retain the null hypothesis.

Approach

We have used grid search with 5 cross validation on each model implemented in order to tune the hyper-parameters (excluding linear regression). The hyper-parameter combinations that return the highest test score will be selected as the final model.

Regression models:

Multiple Linear Regression:

Linear regression is a popular yet naive way to predict continuous variables. The model will fit a straight line that tries to minimise the distance between predicted value and true value. The multiple linear regression will take 21 features into consideration and predict the dependent variable 'point'. Hence, the major parameter the model returns is the intercept and the coefficients of a linear equation. We have set this model as a benchmark because of its simple linear structure.

We have applied *sklearn.linear_model.LinearRegression()* function to build the model. In terms of the model complexity, according to Figure 1 we can see that the training error and generalisation error are very close to each other suggesting our model is not overfitting. The model only takes 0.007s to train, this is because of the low computational complexity for fitting a linear line.

Support Vector Regression:

Support vector machine (SVM) is a popular classification method which finds a hyperplane to separate the data and gives a decision boundary from the plane. Support vector regression (SVR) is based on SVM so that it allows error tolerance. The parameter we have tuned was *kernel* trick and regularisation parameter *C*. We have tested on linear, radial basis function(RBF) and sigmoid kernels which represent different kernel functions that accept inputs from lower dimension and return a dot product to the higher dimensional space(Wilimitis,2018). *C* represents l2 regularisation to the model, which determines how wide the decision boundary is. We have tested on 6 different *C* values. Hence, the grid search tested on 18 different combinations of *kernel* and *C*, returning *kernel = 'rbf'* and *C = 5* as the best combination.
We have applied *sklearn.svm.SVR()* to implement the model. Figure 2 displays the model complexity in which no overfitting is shown. SVR took 0.117s to train which is about 16 times larger than multiple linear regression. This is because SVR's calculation takes place in a higher dimensional space, where the computational complexity will rise significantly.

Random Forest Regression:

Random forest regression is an ensemble learning method which combines predictions from multiple decision trees and forms a more generalised prediction. By using such a method, random forest will not be sensitive to data changes or outlier and it can find the globally optimised result instead of a locally one. We have tuned *criterion* and *n_estimates* parameters in our grid search process. Square error, absolute error and poisson are three different loss functions (*criterion*) that we tested. For *n_estimates*, we tested on 100 and 200 indicating the number of decision trees involved. Generally speaking, more decision trees will give better model performance, however, the computational complexity and risk of overfitting will increase. The best performing parameter combination is *criterion = 'squared_error'* and *n_estimators = 200* from grid search.
We utilised *sklearn.RandomForestRegressor()* to implement the model. The complexity analysis based on Figure 3 suggested that no overfitting is detected. However, random forest regression yields the highest generalisation error across all regression models which may be caused by too many features being learnt. The model takes 0.845s to train, which is the longest among the three regression models due to the calculation on 200 trees.

Model comparison

Hypothesis test is performed according to the set up in the Hypothesis section. From Table 1, both p-values are very small, suggesting there is significant evidence against the null hypothesis and we can conclude that our selected models are different to the benchmark model.

Classification models:

K-Nearest Neighbour:

K-Nearst Neighbour (KNN) is one of the simplest classifiers. It directly stores labels for every data point for the classification instead of performing any complex calculation. Therefore, it is chosen as our benchmark model. The k closest neighour's label will be used to determine the new data point by a majority vote. The performance of KNN highly depends on the k chosen, hence, we have tuned four different *n_neighbors*. A small *n_neighbors* leads to overfitting because it simply takes over the label from the closest point, whereas large *k* leads to underfitting because all the data participates in the majority vote. Another hyperparameter we have tuned is *p* where 1 and 2 represents manhattan distance and euclidean distance. These are two different ways to calculate the distance between points. *n_neighbors = 7* and *p = 1* were the best performing hyper-parameter.
To implement the model, we used *sklearn.neighbors.KNeighborsClassifier()*. The complexity of the model is reflected in Figure 4. Since the gap between training and generalisation error is getting slightly bigger

further to the right side, it suggested a potential overfitting. KNN only took 0.001s to train. This can be explained by the fact that no calculation was done during the training process.

Logistic Regression:

Logistic Regression utilises sigmoid function to map predicted values of probabilities between 0 and 1. With probability bigger than 0.5, the data point will be classified with the corresponding label. The hyper-parameter *C* represents the regularisation term, whereas the smaller the number the more power the regularisation. *solver* determines the different optimization method of loss function. The final combination returned by gridsearch was *C = 5* and *solver = sag*.
sklearn.linear_model.LogisticRegression() is used to implement the model. Complexity graph from Figure 5 suggests no risk of overfitting. Training process took 0.035s which is reasonable for sigmoid calculation.

Support Vector Machine:

As mentioned in the SVR section, Support Vector Machine(SVM) uses a hyper plane to separate hence classify data. Another point with noticing is that its decision boundary for SVM could be non-linear, which could potentially lead to a higher performance. The tuning process can also refer back to SVR tuning. The best parameters chosen for SVM are C = 20 and kernel = 'rbf'.
We used sklearn.svm.SVC() to build the model. Besides, model complexity in Figure 6 shows no sign of overfitting but some oscillation for training error before 500 training samples. It suggests SVM will have better performance with data size bigger than 500. A 0.045s is used for training because of multidimensional calculation again.

Model comparison

The result of the hypothesis test is recorded in Table 2. Since p-value for logistic regression and KNN is larger than the significant threshold 0.05, we obtained the hull hypothesis and concluded that there is no difference between these two models in predicting the pick.

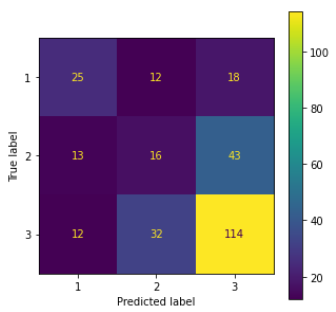
Results

Regression results in predicting the points of the player.

From Table3 in the appendix we can see that the Support vector gives us the smallest standard error which means that at 95% confidence interval we can conclude our prediction range is at most 2.06 points away from the ture point value, which is the closest prediction range amount of three regression models, therefore the most accurate. The reliability wise, the support vector regression gives us the highest r-squared score of 0.953 which means 95.3% of the variance for dependent variable points that is explained by the features (other on court performance statistics) in the support vector regression model. Thus support vector regression gives us not only the best accuracy but also the most reliable one.

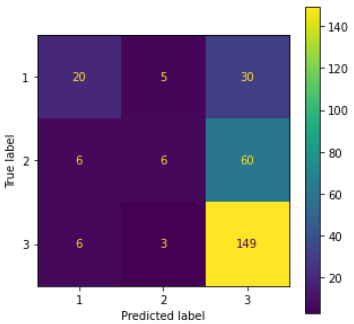
KNN - accuracy on test set:
0.5438596491228070

	precision	recall	f1-score	support
1.0	0.50	0.45	0.48	55
2.0	0.27	0.22	0.24	72
3.0	0.65	0.72	0.68	158
accuracy			0.54	285
macro avg	0.47	0.47	0.47	285
weighted avg	0.53	0.54	0.53	285



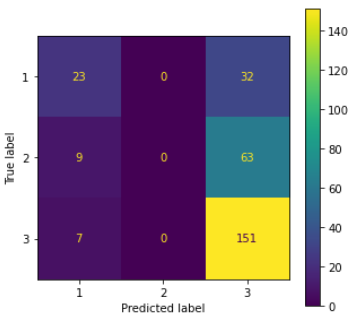
Logistic Regression - accuracy on test set:
0.6140350877192983

	precision	recall	f1-score	support
1.0	0.62	0.36	0.46	55
2.0	0.43	0.08	0.14	72
3.0	0.62	0.94	0.75	158
accuracy			0.61	285
macro avg	0.56	0.46	0.45	285
weighted avg	0.57	0.61	0.54	285



Support vector machine - accuracy on test set:
0.6105263157894737

	precision	recall	f1-score	support
1.0	0.59	0.42	0.49	55
2.0	0.00	0.00	0.00	72
3.0	0.61	0.96	0.75	158
accuracy			0.61	285
macro avg	0.40	0.46	0.41	285
weighted avg	0.45	0.61	0.51	285

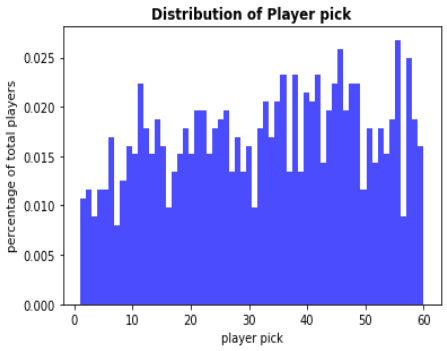


The results from our best model support vector regression indicate that it is a relitivility good model to complete our prediction of points if we are given a players other 20 on court performance statistics.

Classification results in predicting players' pick range.

The performance of these three models by the classification report (class 1 is player pick range 1-14, class 2 is player pick range 15-30, class 2 is player pick range 30-60). As shown in the classification report and the confusion matrix we can see that the logistic regression model gives us slightly higher f1-score in class 2 and class 3 and slightly higher accuracy score than the support vector machine. With the support vector machine fails to predict any class 2 label correctly.we conclude that the logistic regression is a slightly better model here. We also found that all three of the models **did not** do well on predicting the class 2 label with knn performing the best with a 0.24 f1-score.

The classifier is not performing well on predicting the pick range and we conclude 2 reasons on why this is the case. First is that our train dataset is not balanced with less players in the class 2 where pick is between 15-30 and the class 1 where players are in the range of 1-14 as we can see from the following chart.



This reflects on our prediction where class 1 and 2 have worse f1-score than class 3 on all three models. Second reason is that the player pick did not only relate to a player's on court performance, there exist multiple factors outside of the court, for example the ncaa exist 4 regions (East,South,North,West) and these four regions have different level of competition, therefore a player did not get great on court statistic does not necessary means he is not a good player and the NBA teams making decisions on picking the player will consider this fact. Another example will be teams picking players based on their physical potential, players with great height, strength, jumping ability and longer wingspan will more likely to be picked higher and this would not be reflected on the on court performance and thus hinder our models performance.

Conclusion

In conclusion, our two models performed quite differently on two questions. The Support Vector Regression on predicting points gives us great accuracy with high r-square score thus very reliable. We would recommend using this model to do points prediction if having the players on court statistics other than points. Further implementation on different dataset such as NBA player statistics could bring us more insight on whether this model is reliable on different league's players on court statistics. For the classification model predicting the pick, the performance is comparatively low which cannot be utilised as a reliable model. Since the pick is too objective and human behaviours are sometimes illogical, it is hard for any machine learning models to correctly predict. For future improvement, we can try to feed the data into some deep learning neural network and see if the prediction accuracy gets any higher. Throughout this assignment we learn how to implement some machine learning algorithms and get a better understanding of the models and the embedded parameter selections. These would help us to better prepare for new challenges in the future.

Reference

NCAA. (2020). Men's Basketball:Probability of competing beyond high school. Retrieved from https://www.ncaa.org/sports/2015/3/6/men-s-basketball-probability-of-competing-beyond-high-school.aspx

Wilimitis,D. (2018) .The Kernel Trick in Support Vector Classification. Retrieved from https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

Appendix

The dataset we used is available at:
https://www.kaggle.com/datasets/adityak2003/college-basketball-players-20092021

1.Model Complexity Graph:

Figure 1:

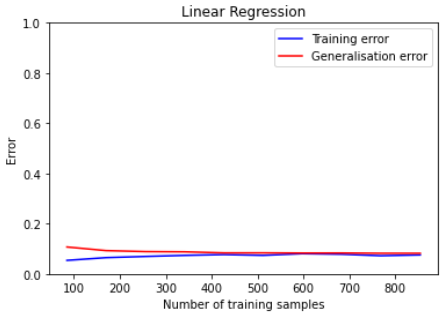


Figure 2:

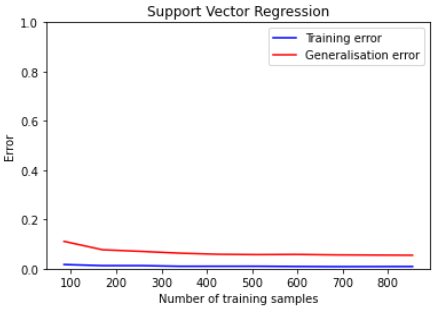


Figure 3:

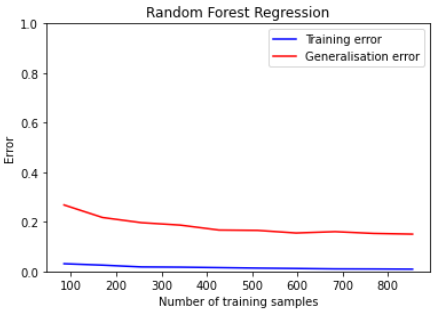


Figure 4:

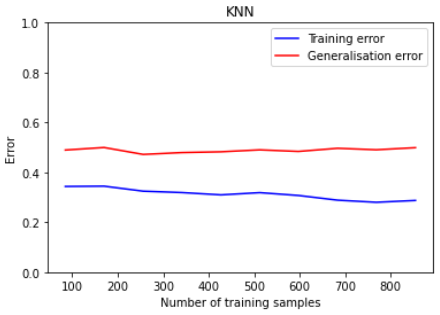


Figure 5:

Figure 6:

2. Hypothesis Test Result Table

Table 1:

Models	**P-value**
Linear Regression & Support Vector Regression	1.55e-8
Linear Regression & Random Forest Regression	8.14e-9

Table 2:

Models	**P-value**
K-Nearest Neighbor & Logistic Regression	0.36
K-Nearest Neighbor & Support Vector Machine	0.01

3.Regression Model Comparison

Table 3:

	Metrics	Range of prediction
Linear Regression	Mean Absolute Error: 0.9703581674098598 Mean Squared Error: 1.8112963367694779 R Squared: 0.9200494896551791 Standard error: 1.370601737000835	within 2.74 points at 95% confidence according to training data. within 2.69 points at 95% confidence according to test data.
Support Vector Regression	Mean Absolute Error: 0.6993928996814971 Mean Squared Error: 1.0645194086151788 R Squared: 0.9530121779285737 Standard error: 0.5972061991729953	within 1.19 points at 95% confidence according to training data. within 2.06 points at 95% confidence according to test data.
RandomForest Regression	Mean Absolute Error: 1.2657502070175437 Mean Squared Error: 2.8958006375499044 R2 score: 0.8721795356568218 Standard error: 0.6597733407182375	within 1.32 points at 95% confidence according to training data. within 3.40 points at 95% confidence according to test data.