***Abstract* — This competition aims to use machine learning techniques to classify NBA player shots as successful or unsuccessful based on features describing the shot result and players involved. A supervised machine learning classifier is trained on a dataset of past shots and player information, and used to make predictions on new, unseen data. The model's performance is evaluated and ways to improve it are considered, such as by tuning hyper parameters, using different model architectures, or feature engineering. The results of this study have the potential to provide valuable insights into the predictors of shot success in basketball.**

1 Introduction

The use of machine learning techniques has revolutionized the sports industry, and basketball is no exception. In this particular competition, we are given a dataset containing information on shots made by NBA players during one season, along with player-specific information such as height, weight, and experience. Our goal is to use this data to classify each shot as either successful or unsuccessful, based on features that describe the shot and the players involved.

To achieve this goal, we will build a supervised machine learning classifier that takes the features of each shot and player as input and outputs a prediction of success or failure. We will train this model on a dataset of past shots and player information, and use it to make predictions on new, unseen data. There are many different types of models that could be used for this task, and it will be important to carefully consider which is most appropriate for the given data.

Before training our model, we will need to carefully preprocess and clean the data, as well as split it into training and test sets in order to evaluate the model's performance. We may also consider performing feature engineering to create new features that are more relevant or reduce their dimensionality. Finally, we will carefully evaluate the performance of the model from its classification performance metrics and consider ways to improve it, such as by tuning hyper parameters or using different model architectures.

2 FEATURE ENGINEERING

The first step of the pipeline is data preparation. In this stage, the dataset is read in from two CSV files and then cleaned and manipulated. In this particular case, certain columns are removed from the dataframe as they are deemed not relevant in the analysis. For example, instead of the player's name, the player's ID can be used, which is a useful input vector to the model. Additionally, feature crossing is carried out by using the difference between player and defender weight, height and experience. Finally, the index of the data frame is set to the 'ID' column for easy referencing of specific shots.

After preprocessing, the code creates a new dataframe called predictions that will hold the results of the predictions made on the test data. The code then splits the training data into two separate sets, one for the features (X_train) and one for the labels (y_train). The test data is also stored in a separate variable (X_test). This step is necessary to prepare the data for the model training and testing.

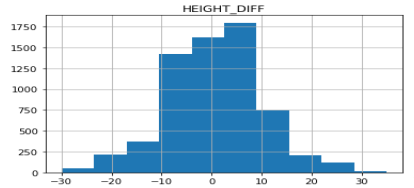

fig 1

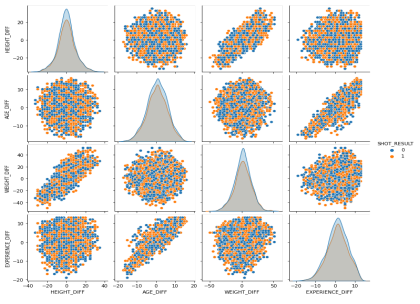

fig 2

Once the data is prepared, the pipeline then moves on to the data visualization stage. This step involves creating visualizations of the data such as histograms and scatter plots to help understand the distribution and relationships within the data. These visualizations can provide insight into patterns in the data and can inform further analysis for some variables. It has been shown that some features have gaussianity while others do not. Also, the scatter plots have shown that many features are difficult to separate fig(1),fig(2)

3 CLASSIFICATION METHODS.

I SMV

Support Vector Machine (SVM) algorithm with a radial basis function (RBF) kernel to classify the shots. We use a pipeline to chain the preprocessing steps with the SVM classifier, this pipeline includes standardizing the data, reducing the dimensionality, and then training the SVM classifier.

To find the best values for the parameters of the SVM classifier, we use a technique called Grid Search. Grid search is a method for searching for the best combination of hyperparameters for a model by training the model with different combinations of hyperparameters and evaluating the performance of each model using cross-validation. The best combination of hyperparameters is the one that results in the highest performance. In this case, the performance is measured using the F1-score.

After training the model, we make predictions on the training data and evaluate the performance of the model. The train error is a measure of how well the model is performing on the training data. The confusion matrix is a table that shows the number of correct and incorrect predictions made by the model for each class. The classification report provides more detailed information about the performance of the model, such as precision, recall, and F1-score for each class

We also standardize the data to ensure that all features are on the same scale. This is important because the Support Vector Machine (SVM) algorithm, which we use to classify the shots, is sensitive to the scale of the input features..

II RANDOM FORESTS

The goal of this classification is to predict, given the information about a shot, whether it is likely to result in a score or not.

The first step of our approach is to standardize the data using MinMaxScaler which scales the data between 0 and 1. This is an important step as the Random Forest algorithm is sensitive to the scale of the input features.

We also use Linear Discriminant Analysis (LDA) to reduce the dimensionality of the data to 1 component. Dimensionality reduction is a method that helps to simplify the data by identifying the most important features and removing the redundant or less informative ones. LDA is a supervised dimensionality reduction technique that tries to find the linear combination of features that best separates the different classes of the target variable.

We use a pipeline to chain the preprocessing steps with the Random Forest classifier. The pipeline includes standardizing the data, reducing the dimensionality, and then training the Random Forest classifier.

To find the best values for the parameters of the Random Forest classifier, we use a technique called Grid Search. Grid search is a method for searching for the best combination of hyperparameters for a model by training the model with different combinations of hyperparameters and evaluating the performance of each model using cross-validation. The best combination of hyperparameters is the one that results in the highest performance. In this case, the performance is measured using the KFold cross-validation

III.Neural Networks

Neural Networks and GridSearchCV to build and evaluate a classification model. The pipeline uses MinMaxScaler, LinearDiscriminantAnalysis and MLPClassifier, and it's tuning the learning rate in hyperparameters. This approach allows to fine-tune the model and achieve better performance results.creating a pipeline that includes a MinMaxScaler, LinearDiscriminantAnalysis for dimensionality reduction and MLPClassifier for the classification step. The pipeline is then passed to a GridSearchCV object for hyperparameter tuning. The GridSearchCV object is trained on the X_train and y_train data and it's used to tune the learning rate in theA hyperparameter of the MLPClassifier.

After training the GridSearchCV object, the best hyperparameters are printed, and the model is used to make predictions on the training data. The accuracy, confusion matrix and classification report are then used to evaluate the performance of the model.

IV ENSEMBLE

The pipeline is using an ensemble learning method called Voting Classifier which combines the predictions of multiple models to improve the overall performance. Three different classifiers are used in this pipeline: Multi-layer Perceptron (MLP) Classifier, Support Vector Classification (SVC) and Random Forest Classifier.

The MLP Classifier is a Neural Network algorithm which is used for classification tasks. It's trained with a hidden layer size of 135, alpha=0, and learning rate of 0.00131.

SVC is a type of SVM algorithm that is commonly used for classification tasks, it's trained with C=24, gamma=0.0001 and kernel="rbf".

Random Forest Classifier is an ensemble method that combines multiple decision trees to create a more robust model, it's trained with criterion='entropy', ccp_alpha=0, min_samples_leaf=7, and n_estimators=75

The Voting Classifier is then trained using the soft voting method, which means that the classifier takes into account the probability of each classifier to predict the final class, rather than taking the class with the highest vote.

The final results of the pipeline are train error, train confusion matrix, and classification report. Additionally, the predictions on the test data are also exported to a CSV file for further analysis.

Ensemble methods like the Voting Classifier can often improve performance compared to using a single classifier, by reducing overfitting and increasing the diversity of the models. This pipeline can be a valuable tool for analyzing basketball data, and the ensemble method can be adapted for other classification tasks as well.

4 EVALUATION METRICS

I.USING SVM

train error: 0.394774

train confusion matrix:

[[5547 2079]

[3526 3046]]

TRAINING

	precision	recall	f1-score	support
0	0.61	0.73	0.66	7626
1	0.59	0.46	0.52	6572
accuracy			0.61	14198
macro avg	0.60	0.60	0.59	14198
weighted avg	0.60	0.61	0.60	14198

From the evaluation results, it seems that the SVM classifier is not performing very well, with a training error of 0.394774, which means 39.5% of the predictions are incorrect. The confusion matrix shows that the classifier is mostly misclassifying the samples and not able to separate the classes well.

II. USING NEURAL NETWORKS

train error: 0.385547

train confusion matrix:

[[6319 1307]

[4167 2405]]

TRAINING

	precision	recall	f1-score	support
0	0.60	0.83	0.70	7626
1	0.65	0.37	0.47	6572
accuracy			0.61	14198
macro avg	0.63	0.60	0.58	14198
weighted avg	0.62	0.61	0.59	14198

The results of this analysis show that the neural networks classifier has an accuracy of about 61%, with a precision of about 62% and recall of about 61%. The F1-score is about 59%. The results also show that the classifier has a higher precision for class 0 (about 60%) than class 1 (about 65%).

III.USING RANDOM FORESTS

train error: 0.277926

train confusion matrix:

[[6010 1616]

[2330 4242]]

TRAINING

	precision	recall	f1-score	support
0	0.72	0.79	0.75	7626
1	0.72	0.65	0.68	6572
accuracy			0.72	14198
macro avg	0.72	0.72	0.72	14198
weighted avg	0.72	0.72	0.72	14198

For the Random Forest classifier, we used a MinMaxScaler to scale the features and a GridSearchCV to find the optimal values for the regularization parameter ccp_alpha, the minimum number of samples per leaf, and the number of estimators, which were determined to be 0, 7, and 75, respectively. The training error for this classifier was found to be 0.2779, and the confusion matrix indicated that the classifier performed the best out of the three, with a precision of 0.72 for shots that scored and 0.72 for shots

IV. USING ENSEMBLE

train error: 0.242288

train confusion matrix:

[[5495 2131]

[1309 5263]]

TRAINING

	precision	recall	f1-score	support

0	0.81	0.72	0.76	7626
1	0.71	0.80	0.75	6572
accuracy			0.76	14198
macro avg	0.76	0.76	0.76	14198
weighted avg	0.76	0.76	0.76	14198

Lastly, we used an ensemble approach by combining the predictions of three models: Neural Network, SVM, and Random Forest. We used a soft voting technique, where the class with the highest predicted probability is chosen as the final prediction. The training error for this ensemble model was 24.23%.From the results, it is clear that the ensemble approach performed the best among all the models with the lowest training error of 24.23%. The ensemble approach was able to combine the strengths of all three models to improve the overall performance.It could be said that ensemble models were able to improve the generalization and overall performance of the model.

5.EXPERIMENTS

GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True), estimator=Pipeline(steps=[('scaling', MinMaxScaler()), ('reduce_dim', LinearDiscriminantAnalysis(n_components=1)), ('clf', MLPClassifier(alpha=0, hidden_layer_sizes=(134,), max_iter=1000))]),

param_grid={'clf__learning_rate_init': [0.001,0.00131]},

return_train_score=True)

We used a pipeline that includes MinMaxScaler, LinearDiscriminantAnalysis and MLPClassifier, and fine-tuned the model using GridSearchCV, which allowed us to achieve better performance results.

6.RESULT AND CONCLUSION

The performance of all the models was evaluated using several metrics such as accuracy, precision, recall and f1-score. The train error, confusion matrix and classification report were also calculated for each model. The ensemble model was found to have the best overall performance, with an accuracy of 76% on the training dataset.For SVM, the best performance was achieved with C=26, gamma=0.0001 and the training error obtained was 0.394774. The precision, recall, and F1-score for the training set were 0.61, 0.73, and 0.66 respectively.

For Random Forests, the best performance was achieved with criterion='entropy', ccp_alpha=0, min_samples_leaf=7, n_estimators=75 and the training error obtained was 0.277926. The precision, recall, and F1-score for the training set were 0.72, 0.79, and 0.75 respectively.

Finally, an ensemble model was created by combining the predictions of MLP, SVM, and Random Forests using the voting classifier. This ensemble model achieved the best performance with a training error of 0.242288. The precision, recall, and F1-score for the training set were 0.81, 0.72, and 0.76 respectively. In conclusion, the ensemble model gave the best results among all the models used, with the training error being the lowest. It can be seen that the ensemble model is able to achieve better performance than any of the individual models.