# March Madness Game Predictor

Jacob Vogel, Izak Little, and Jack Lay

## Part II – Project Identification

Use the following website as inspiration for choosing a project. However, we aren't going to use their data or algorithms, we're better than that! Include in your lab report:

• A summary of your project
• What motivated you to choose this project
• Where the data you used came from

### Project Summary

In this project, we worked to create a machine learning algorithm that predicts the outcomes of the annual NCAA March Madness tournament. To train the algorithm, we used various statistics, such as offensive and defensive percentages, effective field goal percentage, seasonal win rate, and other factors like seed and year. Our goal was to make the algorithm accurate to help make more informed predictions, and stay ahead of the odds.

### Project Motivation

Our project aimed to develop a machine learning algorithm that could accurately predict the outcomes of the NCAA March Madness tournament. To achieve this, we analyzed a range of data sources, including previous seasonal and tournament data, as well as offensive and defensive metrics from Ken Pomeroy and Bart Torvik. We examined several statistics such as seasonal win rate, 2-point and 3-point shooting percentages, effective field goal percentage (a shooting percentage that accounts for the added value of 3-pointers), adjusted offensive and defensive efficiency ratings (the number of points a team scores or allows per 100 possessions, adjusted for the quality of the opponent), turnover rates (the number of turnovers a team has per 100 possessions), and more. We also incorporated key variables like seed (a team's ranking in the tournament) and year to enhance the algorithm's predictions. Our goal was to create a comprehensive and reliable machine learning model that could help NCAA March Madness fans make informed predictions and stay ahead .

# Part III – Algorithm Implementation

Use scikit-learn, Keras, TensorFlow, or another existing implementation to complete your project. In your lab report include:

• Which algorithm you used and why you chose it over alternatives

• Which implementation of the algorithm you used and why you chose it (scikit-learn, TensorFlow, etc.)

## Implementation of Algorithms

We employed Scikit Learn for algorithm implementation of SVM and Gaussian Naïve Bayes models. Our use of Scikit Learn enabled us to set up a pipeline that simplified the process of retraining different models, while also providing consistent data cleaning, normalization, and reshaping during variations of model training.

Initially, we used an SVM to build our model due to the label being binary and continuous features with high dimension. However, the accuracy of the model was unsatisfactory for two reasons. Firstly, we had a relatively small dataset consisting of only 500 games (datums), as we extracted tournament games from 2013 to 2019. Secondly, the high-dimensionality of our data made it challenging to determine if the data was linearly separable.

In order to enable SVM to function effectively, one potential approach is to utilize a kernel to transform the data into a higher-dimensional space where it becomes linearly separable. However, to determine the right kernel was difficult. One strategy for increasing the amount of data (seen in a research paper and data source) was to reorder the features from team 1 to team 2 order and invert/negate features that relate to both teams, while keeping the same label for each game. This approach would have doubled the amount of data available. The research paper discusses how the initial attempt at modeling the data using an SVM was not accurate.

We opted for a Gaussian Naïve Bayes model as it is suitable for limited data. This model excels in handling continuous features, categorical labels, and small datasets. It was a good fit for our data and delivered excellent results. According to the research paper, the Gaussian Naïve Bayes model was determined to be the optimal model for analyzing college basketball tournament data. Additionally, implementing the technique of doubling our available data could further enhance our model's performance. This is particularly relevant as there were fewer games categorized under Team 2, which negatively impacted the accuracy, recall, and f1 score of predicting Team 2 wins.

## Best Algorithm

We achieved the best results with the Gaussian Naïve Bayes algorithm, which involved filtering out features with a pearson correlation coefficient that did not deviate by a certain fraction of the mean correlation coefficient. This filtering was performed separately for positively and negatively correlated features, and also excluded any categorical features from the dataset. Refer to the hyperparameter tuning section to see our best model’s performance scores.

# Part IV – Hyperparameter Tuning

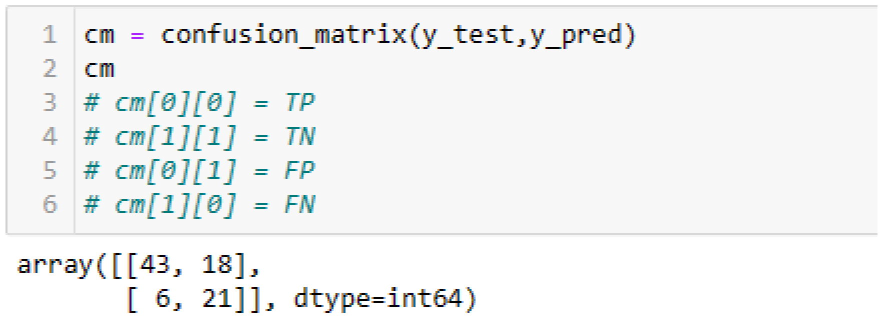
Grid or binary search for hyperparameter tuning is a popular and well researched method for hyperparameter optimization but let's get familiar with other methods. There are a variety of tools to use for tuning, and here are a handful:

• HyperOpt for scikit-learn algorithms
• Keras tuner for Neural Networks

In your lab report include:

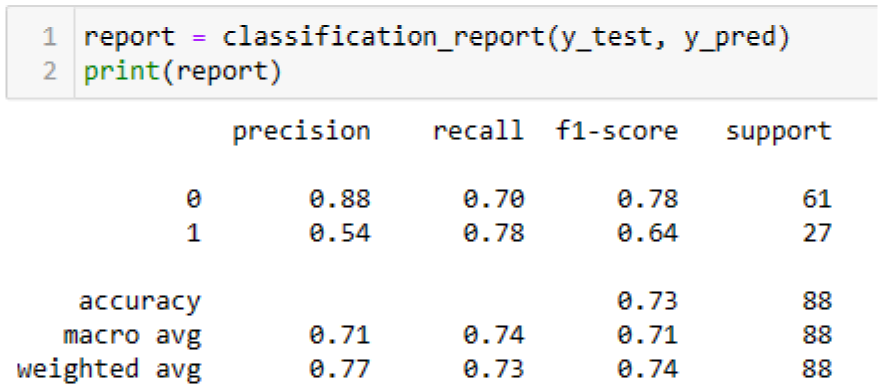
• What tuner you used
• How it works
• A plot comparing your training error using grid search vs your new tuner of choice

As we selected SVM and Gaussian Naïve Bayes as our preferred models, we had limited hyperparameters to fine-tune using methods like grid search. We decided against exploring the possibility of tuning a kernel as SVM did not appear to be a feasible option.For Gaussian Naïve Bayes, we did not have priors and the other defaults for hyper parameters seemed most appropriate for our model. With that being said, we do not have graphs showing the error during an automated hyper parameter tuning tool.
We experimented with various combinations and variations of the features to enhance our Gaussian Naïve Bayes model. Initially, we trained the model using all features, which yielded promising scores, as follows:



```
cm = confusion_matrix(y_test,y_pred)
cm
# cm[0][0] = TP
# cm[1][1] = TN
# cm[0][1] = FP
# cm[1][0] = FN
```

```
array([[43, 18],
       [ 6, 21]], dtype=int64)
```



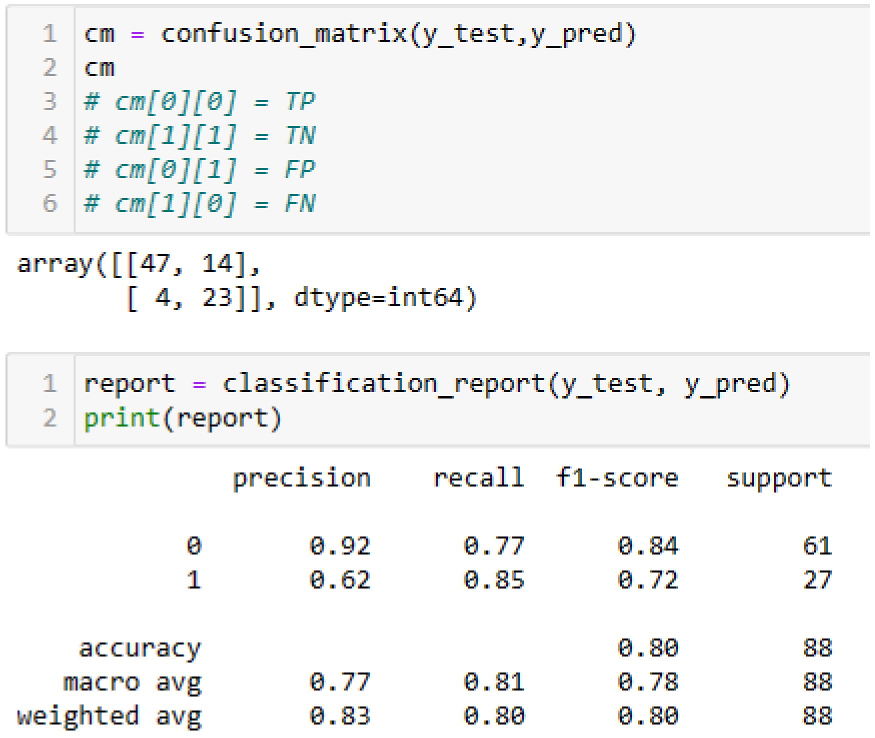
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.88      0.70      0.78        61
           1       0.54      0.78      0.64        27

    accuracy                           0.73        88
   macro avg       0.71      0.74      0.71        88
weighted avg       0.77      0.73      0.74        88
```

Our model was trained using features that had a strong correlation with the label. To select these features, we included those that had a Pearson correlation coefficient outside a fraction of a standard deviation from the mean coefficient. For positively correlated features, we kept those with a correlation coefficient above the positively correlated mean plus a fraction of one standard deviation. For negatively correlated features, we kept those below the negatively correlated mean plus a fraction of the standard deviation. As a result of this change, improvements in the model performance are reflected in the report below:



```
cm = confusion_matrix(y_test,y_pred)
cm
# cm[0][0] = TP
# cm[1][1] = TN
# cm[0][1] = FP
# cm[1][0] = FN
```

```
array([[47, 14],
       [ 4, 23]], dtype=int64)
```

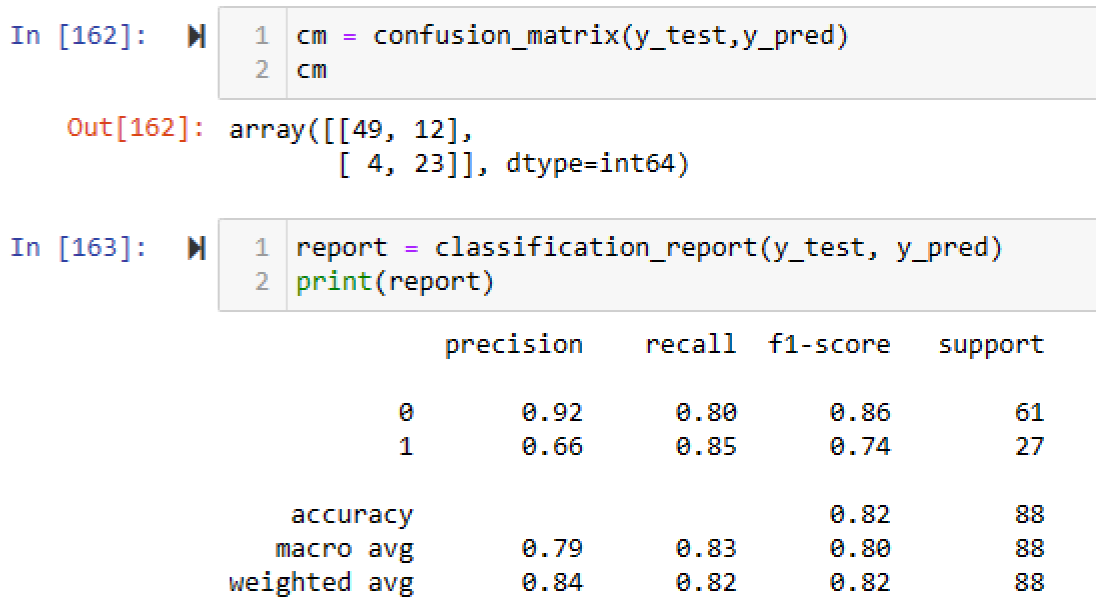
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.77      0.84        61
           1       0.62      0.85      0.72        27

    accuracy                           0.80        88
   macro avg       0.77      0.81      0.78        88
weighted avg       0.83      0.80      0.80        88
```

We made a final adjustment to the features by removing categorical features. This decision was based on Gaussian Naïve Bayes tends to perform better with continuous features and categorical labels. The following report highlights the performance of our top model:



```
In [162]:  cm = confusion_matrix(y_test,y_pred)
            cm

Out[162]: array([[49, 12],
                 [ 4, 23]], dtype=int64)
```

```
In [163]:  report = classification_report(y_test, y_pred)
            print(report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.80      0.86        61
           1       0.66      0.85      0.74        27

    accuracy                           0.82        88
   macro avg       0.79      0.83      0.80        88
weighted avg       0.84      0.82      0.82        88
```

# Part V – Feedback

## Three things we liked about this project

1. We enjoyed the open-endedness of this project. Being able to find something that we are interested in made the tedious processes of finding and cleaning data not as bad and motivated us to create a good dataset and perform good analysis.
2. We also enjoyed working with a team on this project. Having someone to program with helps us figure out problems in our code a lot quicker and helps focus the learning more on the concepts and less on solving annoying bugs.
3. We enjoyed the challenge of finding and combining data to create our final dataset. That ended up being the core learning piece of our project and while it was somewhat cumbersome, once we were able to combine all of our data into a usable set we had a better understanding of both what that process takes and what our data was actually representing.

## Three improvements for next year's class

1. One thing we suggest changing for next year's course would be to have assignments that lead up to this assignment. What is meant by that is that this project accumulates a lot of machine learning concepts outside of algorithms such as data cleaning, data combination, and gaining a general understanding of your data. Some small assignment prior could have granted these skills earlier on to those who maybe haven't practiced it as much.
2. We would also suggest that part IV of this assignment become a bit more generalized. We didn't end up using algorithms with obvious hyperparameters that we could apply to an algorithm such as binary search. A more general approach might be to have part IV be about simply improving the model via whatever methods seem fit.
3. Finally, we all like the idea of using Co-pilot and understand that part I of this assignment was added close to when we received it. However, since it is a paid service, it is probably something that could be left out of next year's assignment. Maybe, as a replacement, the section could be about finding a new tool for machine learning and describe how it gets used and for what purpose.

## General Constructive Feedback

Overall we all found the project to be a good learning experience and good practice for managing and using data to create Machine Learning models. So long as students are motivated and go through the process of defining a problem and looking for machine learning driven solutions they will learn something new in the future. Slight modifications to the project structure should be implemented and tested and we appreciate the opportunity to give feedback.

## Data Sources

1. Sports Reference LLC. (n.d.). College Basketball at Sports-Reference.com. Game by Game Data. https://www.sports-reference.com/cbb/postseason/
2. Kaggle. (2023). March Madness Data. Historical Tournament Data. https://www.kaggle.com/datasets/nishaanamin/march-madness-data?select=Tournament+Team+Data+%28Including+2023%29.csv
    a. KenPom.com. (n.d.). Ken Pomeroy's College Basketball Ratings. https://kenpom.com/
    b. Bart Torvik. (n.d.). BartTorvik.com. https://www.barttorvik.com/#
3. Sundberg, A. (2020). College Basketball Dataset. Historical Season Data https://www.kaggle.com/datasets/andrewsundberg/college-basketball-dataset
4. Akkio. (2021, March 18). Crushing Your March Madness Bracket with AI. https://www.akkio.com/post/crushing-your-march-madness-bracket-with-ai