[GitHub Repository]

Sports statistics have always been a passion of mine, whether it be analyzing quarterback ratings, advanced individual baseball statistics, or measuring the odds of certain events occurring. However, the sport that is nearest and dearest to my heart is college basketball, because of the rivalry and unpredictability of the sport. The dataset I chose for this set detailed the statistics on all NCAA Division 1 basketball teams from the 2012-13 season until the 2022-23 season. (Note that the 2019-20 season was not included with this due to no postseason play.) I wanted to create a model that would classify teams as having the potential to "go deep," or win their first three games, in the NCAA tournament. For reference, the NCAA Tournament is a single elimination tournament that takes 68 teams from across the country, pits them against each other at neutral sites, and over the course of three weekends, a champion is crowned. The ideal model would predict eight teams correctly every year, as only eight teams win at least three games every year in this tournament. To do this, I created a logistic regression model using stratified splits and considered F1 scores of each split to determine the best models.

To begin, the dataset contained 24 unique features and 3523 unique observations. Both numerical and categorical features were present. The majority of numerical variables had to do with average gameplay statistics, with there being specific features for the number of games played, the number of games won, the number of teams they won against that were above the "bubble", the year their season took place, and their NCAA tournament "seeding" position if they had qualified. The categorical variables included the names of the teams, their current conference membership, and their postseason finish if they had qualified.

```
[4]  cbb.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3523 entries, 0 to 3522
Data columns (total 24 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   TEAM        3523 non-null   object
 1   CONF        3523 non-null   object
 2   G           3523 non-null   int64
 3   W           3523 non-null   int64
 4   ADJOE       3523 non-null   float64
 5   ADJDE       3523 non-null   float64
 6   BARTHAG     3523 non-null   float64
 7   EFG_O       3523 non-null   float64
 8   EFG_D       3523 non-null   float64
 9   TOR         3523 non-null   float64
 10  TORD        3523 non-null   float64
 11  ORB         3523 non-null   float64
 12  DRB         3523 non-null   float64
 13  FTR         3523 non-null   float64
 14  FTRD        3523 non-null   float64
 15  2P_O        3523 non-null   float64
 16  2P_D        3523 non-null   float64
 17  3P_O        3523 non-null   float64
 18  3P_D        3523 non-null   float64
 19  ADJ_T       3523 non-null   float64
 20  WAB         3523 non-null   float64
 21  POSTSEASON  680 non-null    object
 22  SEED        680 non-null    float64
 23  YEAR        3523 non-null   int64
dtypes: float64(18), int64(3), object(3)
memory usage: 660.7+ KB
```

Figure 1: All features found in cbb.csv

Cleaning my dataset took longer than expected, primarily due to the fact I wasn't sure what to do with the large number of observations I had. It is important to note that out of all observations, only 680 of them were teams that qualified to play in the NCAA tournament. At first, I wanted to keep them in because I wanted to try and consider all teams, but this skewed the dataset so much that only two percent of teams met the criteria for "going deep." As a result, I adjusted my dataset to contain only teams that qualified for the NCAA tournament. I also went on to create two new variables, "DEEP" and "P7", from existing features present in the dataset. "DEEP", or our target variable, was a Boolean feature that indicated whether or not a team made it to the Elite 8 or further in the postseason. "P7" was a Boolean feature as well that indicated whether or not a team was from a "Power" conference or not. These conferences tend to have more funding and resources than your average Division 1 school, so they typically are more likely to make the postseason and to perform well in the postseason. To make both of these features, I utilized the .isin() function alongside the feature I was drawing my information from to extract the data I needed. After eliminating a few features I found to be redundant and converting all features to floats, I was left with 19 data features and my target feature.

```
# Extract all observations where 'SEED' is not nan
cbb = cbb.dropna(subset=['SEED'])

cbb['CONF'].unique()
# Power 7 Conferences are ACC, B10, B12, SEC, BE, P12, and Amer
# Wish to create a column indicating if an entry belongs to P7 or not
cbb['P7'] = cbb['CONF'].isin(['ACC', 'B10', 'B12', 'SEC', 'BE', 'P12', 'Amer'])

# My target variable for this project is going to be teams "going deep" in the
# NCAA tournament. To go deep, you need to win at least 3 games, or make it to
# the Elite 8, denoted by E8 in the POSTSEASON feature set.
# I am going to create a feature named DEEP that meets this category.
print(cbb['POSTSEASON'].unique())

# Teams denoted as Champion, 2ND, F4, or E8 meet this category.
cbb['DEEP'] = cbb['POSTSEASON'].isin(['Champions', '2ND', 'F4', 'E8'])
```

Figure 2: Cleaning methods used to tidy up and expand the dataset.

```
# Removing TEAM feature, as supposed to view these as blind resumes
cbb = cbb.drop(['TEAM', 'CONF', 'YEAR', 'POSTSEASON', 'G'], axis=1)

cbb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 680 entries, 0 to 3227
Data columns (total 21 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   W        680 non-null    int64
 1   ADJOE    680 non-null    float64
 2   ADJDE    680 non-null    float64
 3   BARTHAG  680 non-null    float64
 4   EFG_O    680 non-null    float64
 5   EFG_D    680 non-null    float64
 6   TOR      680 non-null    float64
 7   TORD     680 non-null    float64
 8   ORB      680 non-null    float64
 9   DRB      680 non-null    float64
 10  FTR      680 non-null    float64
 11  FTRD     680 non-null    float64
 12  2P_O     680 non-null    float64
 13  2P_D     680 non-null    float64
 14  3P_O     680 non-null    float64
 15  3P_D     680 non-null    float64
 16  ADJ_T    680 non-null    float64
 17  WAB      680 non-null    float64
 18  SEED     680 non-null    float64
 19  P7       680 non-null    bool
 20  DEEP     680 non-null    bool
dtypes: bool(2), float64(18), int64(1)
memory usage: 107.6 KB
```

Figure 3: Final dataset contents upon EDA completion.

As I had only two options that could be present for my feature, I sought to treat this as a classification problem. I wanted to exploit thresholds in the features that the majority of qualifying teams would exceed. Before modeling the dataset, I tried to find which features were highly correlated with the target feature. No features had an extremely high correlation, so I had to lower the correlation threshold down to 0.25 to procure more than five features. The six features I had left were the number of wins the team earned, their adjusted offensive efficiency per 100 possessions, their defensive efficiency per 100 possessions, their BARTHAG (which represents the percentage they could win against an average team on a neutral court), the number of wins above bubble they earned, and their tournament seeding.

```
[7] corr = cbb.corr(method='pearson')

    high_corr_features = corr.index[abs(corr['DEEP']) > 0.25]
    print(high_corr_features)

Index(['W', 'ADJOE', 'ADJDE', 'BARTHAG', 'WAB', 'SEED', 'DEEP'], dtype='object')
```

Figure 4: Features that met the 0.25 threshold to be considered in the model.

I wanted at first to use a decision tree setup to model whether these tournament teams could go deep or not. However, as I began to try and apply the model, I discovered this was not a dataset I could apply this onto. The main reason for this was that the NCAA Tournament is, by nature, incredibly unpredictable. Many high seeded teams lose games early on, eliminating them from the tournament. That is what makes this tournament so interesting to the masses, the prospect a team from Nowhereville, Statescatchewan, can knock off a big-name power conference team like Kansas, Kentucky, or Duke. As a result, I chose not to document this attempt and refocused on creating a regression model. I chose to use logistic regression for this model, as teams that performed better in the tournament usually had their results clustered closer together than the ones that didn't.

It is important to note that with this dataset, even after I removed teams that did not qualify for the tournament, there was still a high level of weight shifted towards teams that didn't go deep, with 88.2% of teams not meeting this criterion. As a result, for this model to work, I had to use stratified splits to evenly distribute the teams into the train and test sets used for modeling. To find the results of interest, I considered three criteria to see which best result of these three produced the most accurate results. Within each k-split for loop, I chose to examine the results of each model produced and store the best one. I was then able to retrain the stored model as my final model. To see this, please refer to code block 10 in azinkSelectionAndTraining.ipynb, which can be found in my GitHub repository for this project.

To test the model's performance, I chose to use another dataset from within the original files. This set was from the previous season (2023-24), but was incomplete, as it was published before the NCAA Tournament was completed, therefore having no target variable. Since the results of this tournament are in the public eye now, I chose to use this set as my evaluation set, rather than generate random observations.

```
cbb24 = pd.read_csv('cbb24.csv')

print(cbb24.head(10))
```

```
   RK            TEAM CONF   G   W  ADJOE  ADJDE  BARTHAG  EFG%  EFGD%  ...  \
0   1         Houston  B12  34  30  119.2   85.5   0.9785  49.7   44.0  ...
1   2     Connecticut   BE  34  31  127.1   93.6   0.9712  57.1   45.1  ...
2   3          Purdue  B10  33  29  126.2   94.7   0.9644  56.0   47.7  ...
3   4        Iowa St.  B12  34  27  113.6   86.5   0.9583  51.9   47.1  ...
4   5          Auburn  SEC  34  27  120.7   92.1   0.9573  54.1   43.4  ...
5   6         Arizona  P12  33  25  121.5   93.6   0.9526  55.0   48.7  ...
6   7       Tennessee  SEC  32  24  115.6   91.2   0.9382  51.5   45.4  ...
7   8       Marquette   BE  34  25  118.9   94.6   0.9328  55.1   49.7  ...
8   9  North Carolina  ACC  34  27  116.8   93.2   0.9305  51.3   46.4  ...
9  10       Creighton   BE  32  23  120.6   96.5   0.9289  57.5   46.4  ...

    DRB   FTR  FTRD  2P_O  2P_D  3P_O  3P_D  ADJ_T   WAB  SEED
0  30.2  29.9  39.0  48.4  43.4  34.7  30.0   63.3  10.6   1.0
1  26.8  33.3  32.5  58.5  43.7  36.7  31.9   64.6  11.3   1.0
2  24.7  42.8  23.0  53.2  48.1  40.8  31.4   67.6  11.0   1.0
3  29.4  36.1  35.2  51.7  46.9  34.9  31.5   67.6   6.9   2.0
4  30.3  38.2  41.0  54.9  42.8  35.2  29.8   69.8   5.5   4.0
5  23.1  36.7  25.7  54.6  47.8  37.1  33.4   72.0   4.9   2.0
6  28.6  34.3  35.8  51.6  44.3  34.2  31.4   69.2   5.8   2.0
7  30.8  25.6  28.2  56.0  49.2  35.8  33.6   69.1   6.5   2.0
8  23.5  36.8  28.3  50.3  46.0  35.4  31.4   70.4   6.6   1.0
9  24.6  24.5  16.2  60.6  45.7  36.1  32.2   66.9   4.4   3.0
```

Figure 5: A sample of the first 10 observations from the cbb24 dataset.

For all three sets, the dataset was underfit. There were less positives than there were teams that actually made it deep (3 positives out of 8) and at least one of them turned out to be a false positive. Prioritizing precision or the F1 score led to the same result: UConn, Houston, and Purdue all had the caliber to go deep. Houston was the false positive in this set, as they lost in the Round of 16, while UConn and Purdue went on to play each other for the national title.

```
test24 = cbb24_final.values
# use the model to predict the outputs
pred24 = model.predict(test24)

print(pred24.shape)
print(pred24)

# Find the number of 1s present in the outputted array
count = np.count_nonzero(pred24 == 1)
print(count, "teams had the potential to go deep.")
```

```
(68,)
[1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
3 teams had the potential to go deep.
```

Figure 6: Output of model prioritizing F1 statistic.

Prioritizing recall, on the other hand, swapped out Purdue with Auburn, who was eliminated in their first game in the tournament. This is not to say that these results are poor, rather that the model had a hard time splitting up teams that did not meet the criteria from those that did. One thing I noticed of the teams that met the criteria in this model is that they most consistently were number 1 seeds, or the best seeded teams in the tournament. They typically have the best odds of advancing the furthest in the tournament, with them being the only seed line that has better than 50% odds of advancing to the Elite 8, according to [the University of Illinois at Urbana-Champaign](#).

To conclude, I believe that producing this model definitely had its challenges. Prioritizing teams that qualified over teams that did not was one that stuck out for me, which was easy to handle with a removal of all teams that possessed an NA for the SEED feature. Trying to optimize Decision

Tree selection was also a struggle as well, as I believed that this would have been a more optimal way to select these teams. I think a decision tree application would be better suited for a playoff structure like what the NBA uses, where teams play against each other over the course of a series of games, with a team having to win 3 or 4 games against their opponent in order to advance. Finally, I would like to think that separating teams that belong to Power conferences and non-power conferences from each other would allow for more accurate predictions. Non-power conference teams typically do not perform as well, but there are a few outliers that lead me to believe they should still be considered in the analysis of this dataset. This could be done by using the P7 feature created in the EDA section of the project.