Forecasting the 2023 NCAA Basketball Tournament

Final Project Presentation | MSDS 565, Spring 2023

20 March 2023

Authors: Aleesa Mann and Cyruss Tsurgeon

Agenda

- Introduction – **Problem Understanding** and Definition
- Dataset and Features – **Data Collection** and Preparation
- Methods – Data Understanding using **EDA**
- Experiment and Results – **Model Building**
- Experiment and Results – **Model Evaluation**
- Conclusions – **Communication** (and Deployment)
- Questions

Introduction

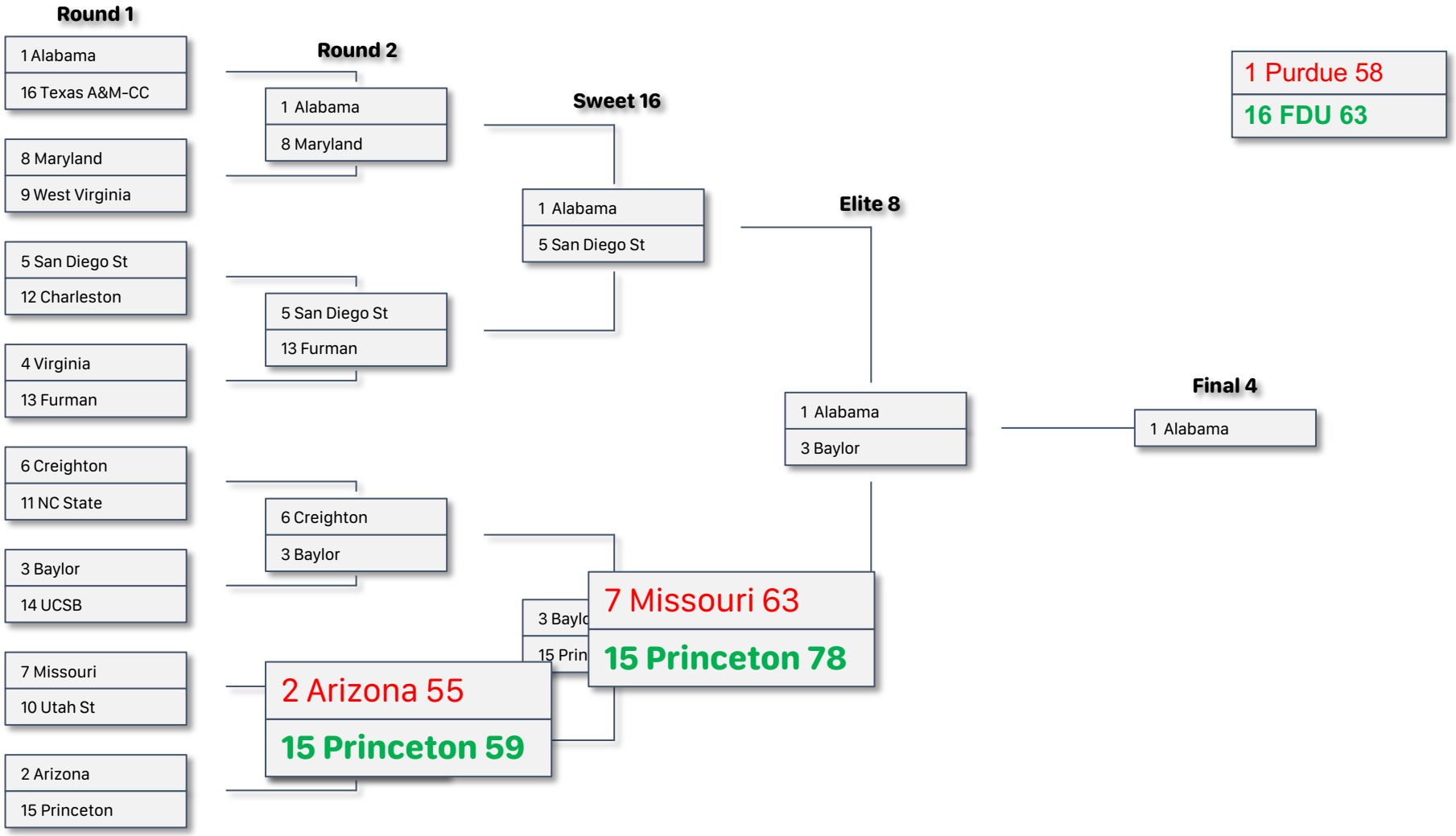
Introduction – Problem Understanding

- **predict the winner** of every individual game of NCAA College Basketball Tournaments
- according to the AGA, up to **70 million brackets** in 2017[1]; an estimated **45 million people** in 2022[2]

Source : [1] David Purdum (2017) | [2] American Gaming Association (2022)

How the brackets work.

How the brackets work.

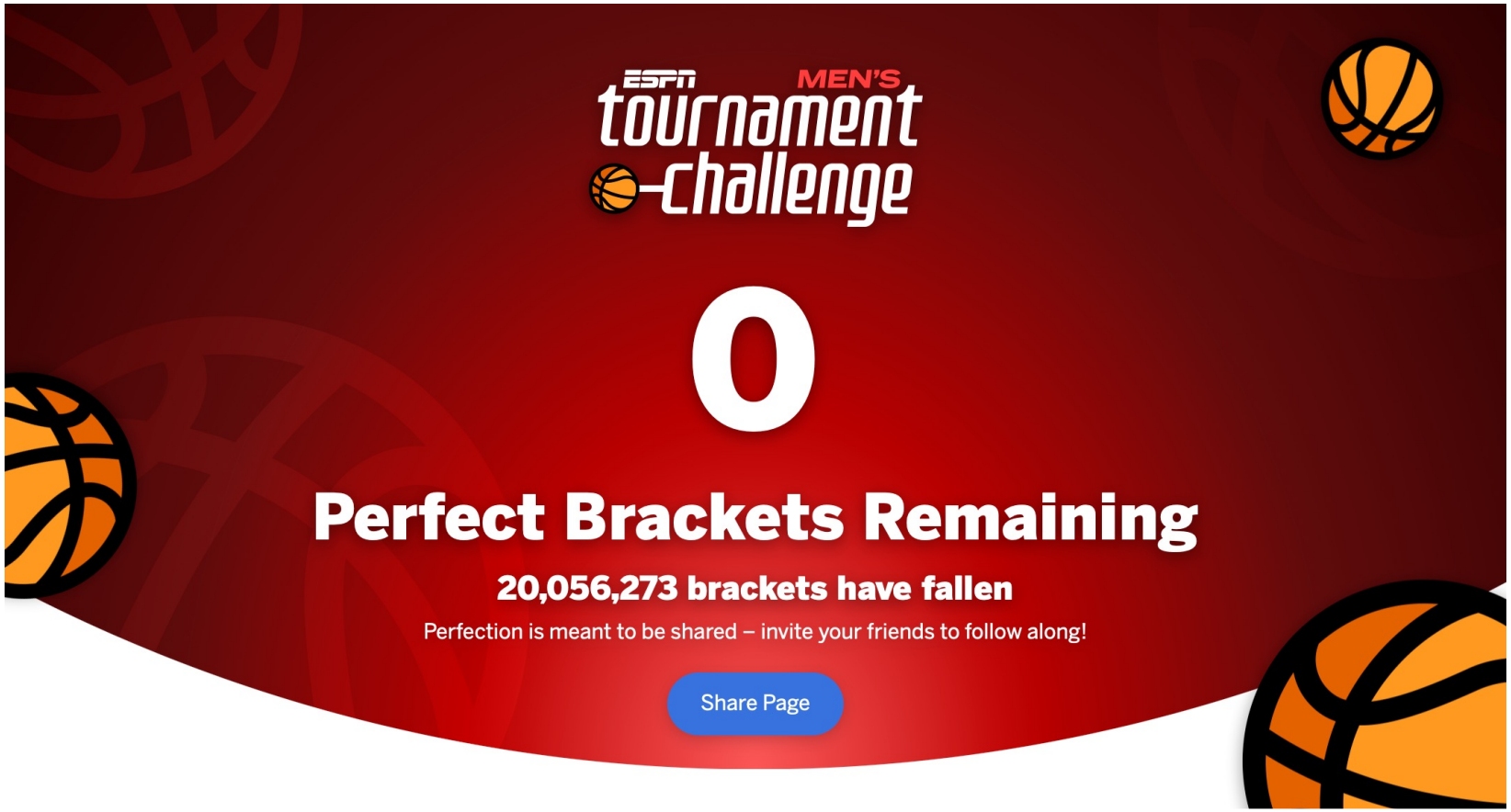


Introduction – Problem Understanding

- **predict the winner** of every individual game of NCAA College Basketball Tournaments
- according to the AGA, up to **70 million brackets** in 2017[1]; an estimated **45 million people** in 2022[2]
- no one has ever completed a **'perfect bracket'**

Source : [1] David Purdum (2017) | [2] American Gaming Association (2022)

Perfect brackets



Introduction – Problem Understanding

- **predict the winner** of every individual game of NCAA College Basketball Tournaments
- according to the AGA, up to **70 million brackets** in 2017[1]; an estimated **45 million people** in 2022[2]
- no one has ever completed a **'perfect bracket'**
- odds of completing a 'perfect bracket' are 2^{63} or 1 in 9,223,372,036, 854,775,808 - **1 in 9.2 quintillion** [3]
- streak of correct bracket picks is at **49 games** [4]
- used provided data and competitive structure of the Kaggle **2023 March Machine Learning Mania** competition[5]

Source : [3] Daniel Wilco (2023) | [4] Brian Budzynski (2022) | [5] J. Sonas, et al (2023)

Introduction – Problem Definition

- millions of brackets predicting Champion for 63 tournament games
- more complex task with an objective to predict the outcome of a hypothetical matchup between each team against all other teams
  - 363 men teams, thus 363 * 362/2 = **65,703** combinations
  - 361 women teams, thus 361 * 360/2 = **64,980** combinations
  - final submission file must have 65,703 + 64,980 = **130,683** predictions
- additional complexity to submit a probability for each outcome

Source : J. Sonas, et al (2023)

Introduction – Problem Definition

- we simplified our problem definition for this project
- we divided our problem into men's and women's predictions for each of us to tackle, and we combined our results.

- **Goal:** Predict the probability of all possible matchup between the 64 teams selected for the College Basketball Tournament (64 * 63/2 = 2,048 predictions)

Source : J. Sonas, et al (2023)

Introduction – Related Work

- in **2012**, Chris Wright provided a statistical analysis of predictors for March Madness [6]
- in **2014**, Alex Tran and Adam Ginzberg reported nearly all of the game statistics were useless except for FG%, FT%, and 3PT% [7]
- in **2014**, Levi Franklin found margin of victory, seeding, and previous tournament performance were useful features [8]
- in **2018**, Cody Kocher and Tim Hoblin found that the correlation between the statistical variables used for analysis tended to be high among several variables[9]

Source : [6] Chris Wright (2012) | [7] Tran and Ginzberg (2014) | [8] Levi Franklin (2014) | [9] Kocher and Hoblin (2018)

Dataset and Features

Dataset and Features – Data Section 1 Main Files

- **Seasons :** Season, DayZero, RegionW, RegionX, Region Y, Region Z
- **Teams :** TeamID, TeamName, FirstD1Season, LastD1Season
- **Regular Season Detailed Results :** Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT, WFGM, WFGA, WFGM3, WFGA3, WFTM, WFTA, WOR, WDR, WAst, WTO, WStl, WBlk, WPF
- **Tourney Detailed Results :** Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT, WFGM, WFGA, WFGM3, WFGA3, WFTM, WFTA, WOR, WDR, WAst, WTO, WStl, WBlk, WPF
- **Tourney Seeds :** Season, Seed, TeamID
- **Sample Submission :** ID, Pred (e.g. 2018_3181_3314,0.516)

Dataset and Features – Data Section 2 Extra Files

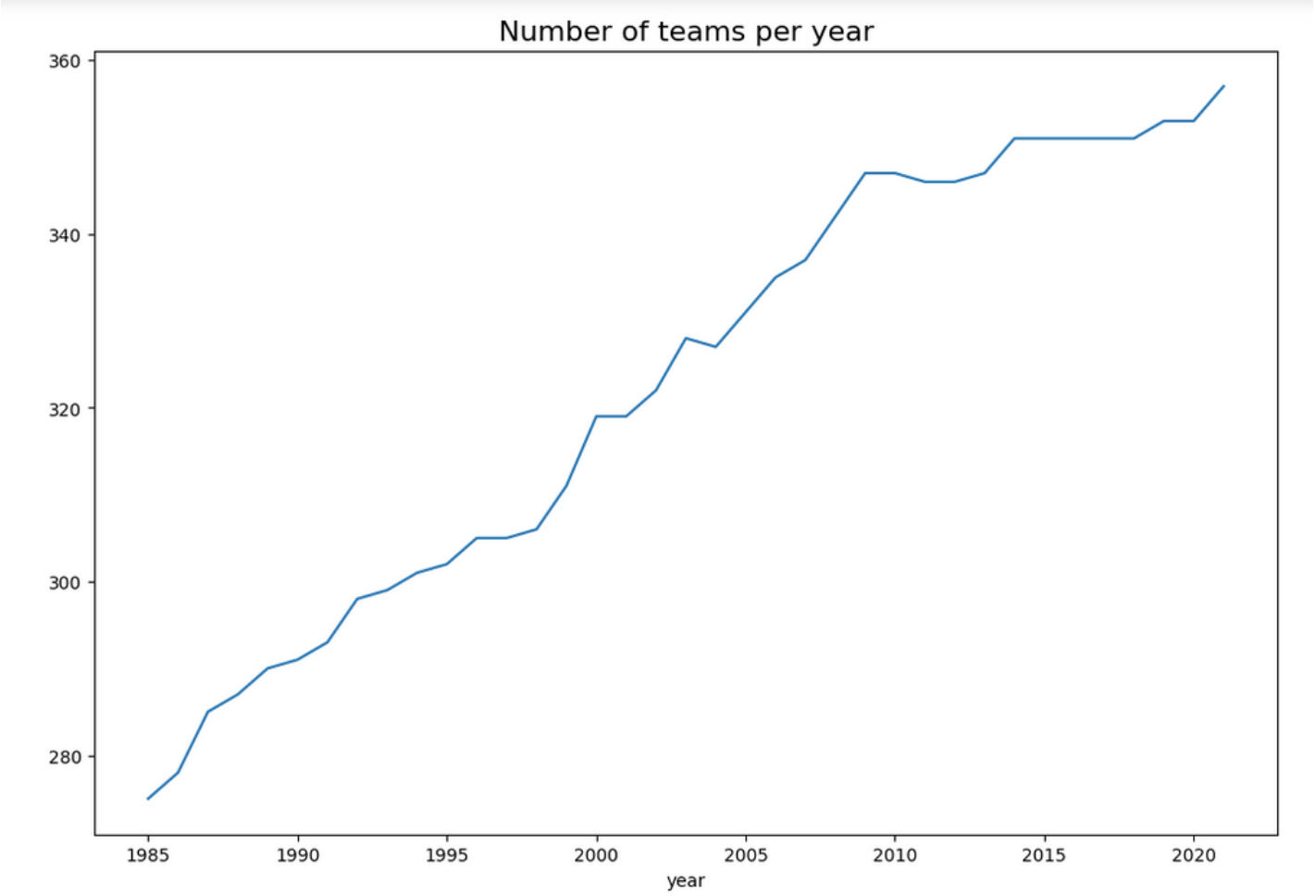
- **Cities :** CityID, City, State
- **Game Cities :** Season, DayNum, WTeamID, LTeamID, CRType, CityID
- **Compact Season Results :** Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT
- **Compact Tourney Results :** Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT
- **Compact Secondary Tourney Results :** Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT, SecondaryTourney
- **Tourney Slots :** Season, Slot, StrongSeed, WeakSeed

Dataset and Features – Data Section 3 Supplements

- **Team Coaches :** Season, TeamID, FirstDayNum, LastDayNum, CoachName
- **Conferences :** ConfAbbrev, Description
- **Team Conferences :** Season, TeamID, ConfAbbrev
- **Conference Tourney Games :** ConfAbbrev, Season, DayNum, WTeamID, LTeamID
- **Secondary Tourney Teams :** Season, SecondaryTourney, TeamID
- **Massey Ordinals :** Season, RankingDayNum, SystemName, TeamID, OrdinalRank
- **Team Spellings :** TeamNameSpelling, TeamID
- **Seed Round Slots :** Seed, GameRound, GameSlot, EarlyDayNum, LateDayNum

Methods

Methods – Data Understanding using EDA



Methods – Data Understanding using EDA

Number of tournament wins by seed (for years 2003 - 2021):

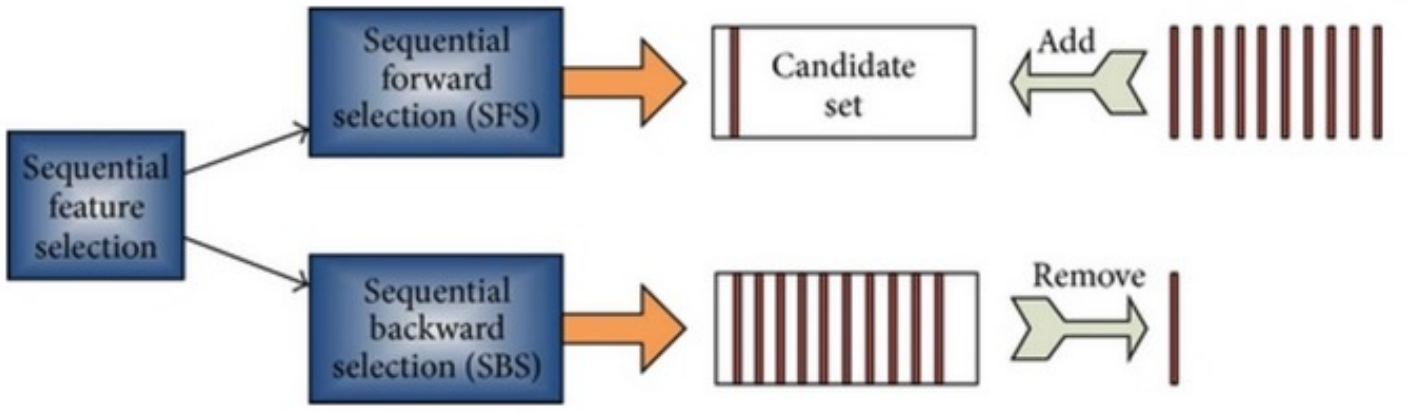
	wincount
1	158
2	108
3	99
4	92
5	83
6	65
7	62
8	64
9	59
10	45
11	81
12	62
13	41
14	33
15	33
16	57

In this dataset, if you look at at the seeds for team_1 and team_0, and you were to guess that team_1 won every time they had a lower seed number (which is a higher ranking) than team_0, then you would be right 33% of the time. We want our models to perform better than that.

	team_1	Seed_1	team_0	Seed_0	winner
0	1411	16	1421	16	0
1	1400	1	1421	16	1
2	1112	1	1436	16	1
3	1112	1	1211	9	1
4	1153	8	1211	9	0
...	...	...	...	...	...
1176	1325	13	1438	4	1
1177	1222	2	1333	12	1
1178	1329	4	1333	12	0
1179	1260	8	1333	12	0
1180	1234	2	1332	7	0

Methods – Data Understanding using EDA

- For feature selection, we used the scikit learn Sequential Feature selector and forward selection and time series cross validation to reduce the feature set.



Two variants of sequential feature selection: the sequential forward selection and sequential backward selection.

Source : [10] Zhang, et al. (2013)

Experiments

Experiment – Testing and Training Split

- Because this is a time series dataset, we had to make sure our training dataset was entered into the model sequentially.

In [222]:
```
def backtest(data, model, predictors, start=2, step=1):
    all_predictions = [] # list odf dataframe, each dataframe is the prediction for a single seasons
    seasons  = sorted(tournaments["season"].unique()) #list of seasons tha texist in data

    for i in range(start, len(seasons), step): #we progress with step parameter (one season at a time)
        season = seasons[i] # pull out the season number / 2013 will be the first season

        train = tournaments[tournaments["season"] < season] #for 2013, training will be 2012 and 2011
        test = tournaments[tournaments['season'] == season] #test will be 2013 (or whatever the current season is)

        model.fit(train[predictors], train['winner']) #this will take training data, all the predictors
        preds = model.predict(test[predictors]) #generate predictions from the test set we want

        #by default model.fit  will return numpy array, but we want to convert it to a pandas series, keep index the
        preds = pd.Series(preds, index=test.index)

        combined = pd.concat([test['winner'], preds], axis=1) #axis=1 means treats target and preds as two columns,
        combined.columns = ['actual', 'predictions'] #renaming columns

        all_predictions.append(combined) #this will be a list of five dataframes, one for each season, then we will

    return pd.concat(all_predictions)
```

Experiment – Model Building

Model	# Features	Tuning
SVM	19	kernel = 'linear', probability = True
KNN	19	k = 5
GNB	19	
DT	19	max_depth = 5
RF	19	n_estimators = 10
MLP	19	alpha = 1, max_iter = 1000

Experiment – Model Building

Built stacked model using 5 or 6 trained models as estimators; and using Logistic Regression as algorithm

```
# Define estimators
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression

estimator_list = [ ('svm', svm), ('knn', knn), ('gnb', gnb), ('rf', rf), ('mlp', mlp) ]

# Build stack model
model = StackingClassifier(estimators=estimator_list, final_estimator=LogisticRegression())
```

Results – Model Evaluation

Model	# Features	Tuning	Accuracy
SVM	15	Linear Kernel	.69
	10	Linear Kernel	.69
	10	RBF Kernel	.71
Ridge Regression	15	Alpha = 1	.70
	15	Alpha = 1	.70
	10	Alpha = 0.01	.69
Logistic Regression	15	Data Scaling	.70
	10	Data Scaling	.70

Results

Results – Model Evaluation

Model	# Features	Tuning	Accuracy
SVM	10	RBF Kernel	.66
Ridge Regression	10	Alpha = 0.01	.69
Logistic Regression	10	Scaling Data	.64

- Smaller feature sets performed better
- All models performed similarly (which may be due to the small data set size)
- SVM with radial basis function kernel, which may be because it can better pick up nonlinearity in data by projecting the data into more dimensions
- Ridge Regression performed best in testing data

Results – Model Evaluation

Scikit Learn Metrics

```
from sklearn.metrics import brier_score_loss, accuracy_score, matthews_corrcoef, f1_score
from keras.metrics import binary_accuracy
from keras import backend as K
```

Brier Score Metric

```
# Define Brier score function
def brier_score(y_true, y_pred):
    return K.mean(K.square(y_true - y_pred), axis=-1)

# Define custom metric function wrapper
def brier_metric(y_true, y_pred):
    return 1 - brier_score(y_true, y_pred)

# Define Brier score function
def brier_score_metric(y_true, y_pred):
    return K.mean(K.square(y_true - y_pred))
```

Results – Model Evaluation

Stacked Model

```
Model performance for Training set
- Brier: 4.087152605165435e-07
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Test set
- Brier: 6.749724411646529e-07
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
```

Summary Table

	Brier	Accuracy	MCC	F1
svm	2.748182e-05	1.000000	1.000000	1.000000
knn	2.823448e-02	0.969051	0.938038	0.969049
gnb	2.758299e-02	0.962047	0.924020	0.962047
rf	4.619970e-04	1.000000	1.000000	1.000000
mlp	2.369699e-04	1.000000	1.000000	1.000000
model	4.087153e-07	1.000000	1.000000	1.000000

Conclusions

Conclusions – Communication (and Deployment)

- Using the **sequential features selector** across the RidgeRegression, Logistic Regression, and SVM models, there were some variables that appeared in more often in the selection processes, in particular, seed_1 - the seed number for team 1 - was used in all eight models.

- This means **this variable has significant weight** in determining the outcome and should be considering in future predictive modeling that undertakes this task of tournament prediction.

Conclusions – Communication (and Deployment)

- There needs to be a way to account for the **randomness** of the matchups, what kind of metric can be used for this? (i.e. a player is hurt, team's lack of sleep due to travel schedule, etc.)

- Especially considering this is a **single-elimination tournament**, which means any aberration in performance could mean a top team losing or a bottom team winning.

- This type of **unpredictability** is the 'madness' of March Madness.

Conclusions – Communication (and Deployment)

- This project was **fun**, but **very challenging**. Trying to accurately predict outcomes that show so much unpredictability proved to be very difficult.

- One learning point from this project was that we should have **provisioned our time at the onset** (to allow sufficient time for model and hyperparameter tuning).

- Another learning point, feature engineering is very important. It would have been better to establish '**hot team**' and '**tired team**' features - for streaking teams; **bring in time-series**.

Conclusions – Communication (and Deployment)

- I think using a **convoluted neural network model** with transfer learning from regular season data might be a better approach.

- The **brier score** metric was new for us, fully understanding this metric and using this metric to optimize model may also prove to be a smarter tactic.

References

1. David Purdum (2017). ESPN – 70 million brackets, $10.4 billion in bets expected for March Madness. American Gaming Association. https://www.americangaming.org/new/espn-70-million-brackets-10-4-billion-in-bets-expected-for-march-madness/ accessed: 18 March 2023.

2. American Gaming Association (2022). 2022 March Madness Wagering Estimates. American Gaming Association. https://www.americangaming.org/resources/march-madness-2022/ accessed: 18 March 2023.

3. Daniel Wilco (2023). The absurd odds of a perfect NCAA bracket. https://www.ncaa.com/news/basketball-men/bracketiq/2023-03-16/perfect-ncaa-bracket-absurd-odds-march-madness-dream accessed: 18 March 2023.

4. Brian Budzynski (2022). Who picked the best March Madness bracket of all time? https://www.wavy.com/sports/ncaa-bastketball/who-picked-the-best-march-madness-bracket-of-all-time/ accessed: 18 March 2023.

5. Jeff Sonas, Last-Place Larry, Maggie, and Will Cukierski (2023). March Machine Learning Mania 2023. Kaggle. https://kaggle.com/competitions/march-machine-learning-mania-2023 accessed: 18 March 2023.

References

6. Chris Wright (2012). Statistical Predictors of March Madness: An Examination of the NCAA Men's' Basketball Championship. Thesis for Pomona College Economics Department.

7. Alex Tran and Adam Ginzberg (2014). Making Sense of the Mayhem: Machine Learning and March Madness. Stanford CS229 Final Project paper December 2014.

8. Levi Franklin (2014). Predicting March Madness: Winning the Office Pool. Stanford CS229 Final Project paper December 2014.

9. Cody Kocher and Tim Hoblin (2018). Predictive Model for the NCAA Men's Basketball Tournament. Ball State Undergraduate Mathematics Exchange vol 12(1) 15-23.

10. Zhang, Yudong & Wang, Shuihua & ji, Genlin. (2013). A Rule-Based Model for Bankruptcy Prediction Based on an Improved Genetic Ant Colony Algorithm. Mathematical Problems in Engineering. 2013. https://www.hindawi.com/journals/mpe/2013/753251/

Questions?