Project 11
Predicting Purdue's Performance in the NCAA Tournament Bracket
Business Understanding Report

Saul Means, Kuba Bal, Sarah Firestone, Zachary Hanson, Avi Khurana
Presentation Effort:
Zachary Hanson 45% Avi Khurana 45% Sarah Firestone 10%
Report Effort:
Saul Means 25% Sarah Firestone 25% Avi Khurana 25% Zachary Hanson 25%

1. SELECT MODELING TECHNIQUE

1.1. Modeling technique

We decided that classification and supervised learning is the best modeling technique for this project. The reasoning behind this decision is that we want to predict outcomes as our final goal so we need to use classification methods and models. Classification and supervised learning are more appropriate than clustering methods in this project since we have labeled data for the past seasons.

This is a classification task. Hence, we will use four algorithms:
1) K Nearest Neighbors (KNN): This method takes a new unknown point and compares it to its neighbors to see what it should be labeled as. This is a simple model that has a strong classification accuracy which seems promising. Also after researching possible models, numerous articles have suggested KNN to be one of the best performing models for this assigned task.

2) Logistic Regression: by using a multinomial approach rather than a one vs all approach. Used a grid search for hyperparameters and calculated feature importance and also used platt scaling for prediction adjustments.

3) SVM: By using grid search for hyperparameter settings, calculated features importances, and platt scaling for prediction adjustments.

4) Random Forest: by using the Gini criterion and a maximum number of trees.

1.2. Modeling assumptions

For K Nearest Neighbors, it is a non-parametric model so it does not assume anything about the training data. It does not build the model from the training data so no training is needed. Although the classification time is linear, the model assumes the data itself does not need to be linear. There are also no assumptions about the distribution of the data as well. The model also can have missing data as it will estimate those

values through its process of missing data imputation, however, this will not need to be used as we have no missing data as seen in our report 2 and 3. KNN does assume the data is numeric as it cannot handle categorical data. Report 2 lists the columns we will use which are numeric, however, some features such as TeamName are categorical so if that is decided as an important feature (after conducting top feature selection) then we will try one hot encoding to transform the variable in the compatible form for KNN.

SVM makes several assumptions about the data. It assumes that the data is separable by at least one hyperplane, that the data can be modeled as linear combinations of the features, that the data is entirely numerical, that the data is standardized, and that the data does not have that much noise.

Logistic regression makes a few assumptions about the data. It assumes that all of the data is numerical, that the data is standardized and that there are no extreme outliers in the data.

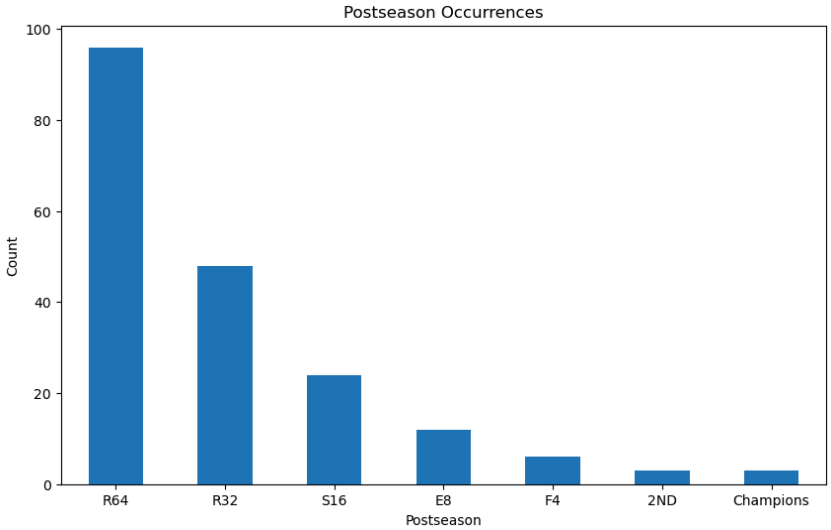
Random forest makes a couple assumptions. It assumes that all of the data is numerical or categorical and that the features are independent of each other (non-highly correlated features).

All of the previous assumptions co-align with the information presented in our data preparation and data understanding reports.

2. GENERATE TEST DESIGN

The data is cleaned and prepared so the next step is to determine which features are the most important for our models. Two different feature importance methods were used: Random Forest, and Feature Importance (ExtraTreesClassification). Both methods confirmed that the top features are, 'AdjEM', 'SEED', 'WAB', 'AdjDE', 'de_TOPct','3P_O', 'de_ORPct', 'OE', 'DE', 'o_TOPct'.

We decided we are likely going to use weights as the class distribution can be seen in this graph:

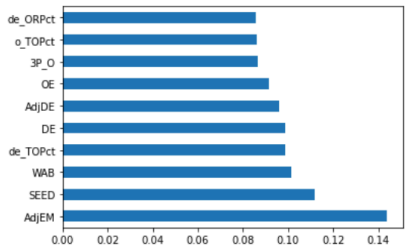


Random Forest:

```
AdjEM        0.096570
SEED         0.062966
WAB          0.057535
AdjDE        0.057180
de_TOPct     0.051726
3P_O         0.048502
de_ORPct     0.046549
OE           0.046232
DE           0.042566
o_TOPct      0.042108
```

Feature Importance (ExtraTreesClassification)

```
[0.14349715 0.11175585 0.10148789 0.09578075 0.09867916 0.08673076
 0.08582672 0.09139092 0.09856196 0.08628883]
```



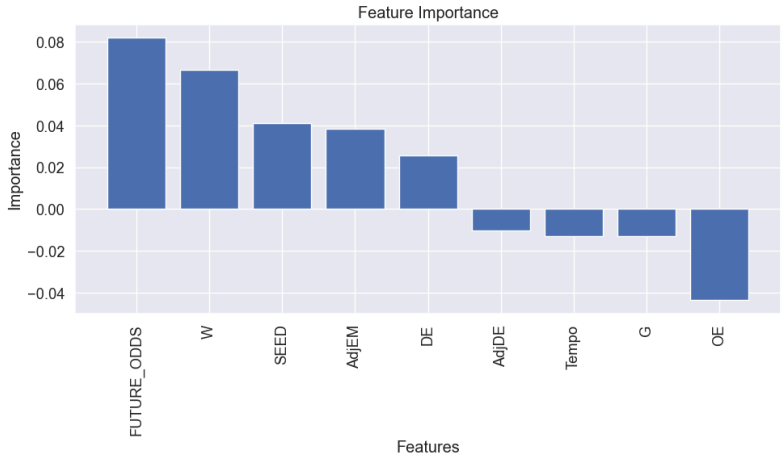
The data was then split into 2 parts: the "20% dataset" (for testing) and "80% dataset" (for feature importance calculating, parameter tuning, and model training). However for KNN, splitting the data by 30% for training was also evaluated and compared. For KNN, the values from 2 -29 (the recommended k value range) were each used to find which was the best k.

The following steps are taken for model testing:

SVM:

1) The dataset is split into 2 parts: the "20% dataset" (for testing) and "80% dataset" (for feature importance calculating, parameter tuning, and model training).

2) feature importance calculations run on all columns in order to exclude columns with negligible importance.



3) Grid search performed on 80% training data in order to find best hyper parameter settings based on "accuracy" metric.

```
Best parameters: {'C': 1, 'coef0': 0, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'}
```

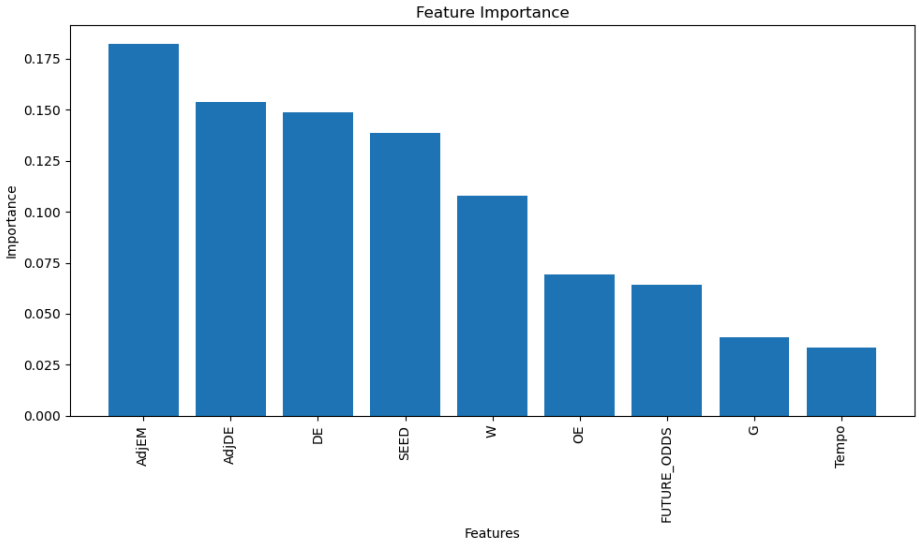
4) The remaining "20% dataset" will be used to test and evaluate the model by calculating

accuracy, precision, recall, and F1-score.

```
Classification Report:
              precision    recall  f1-score   support

         2ND       0.00      0.00      0.00         2
   Champions       0.00      0.00      0.00         1
          E8       0.00      0.00      0.00         1
          F4       0.33      0.33      0.33         3
         R32       0.50      0.56      0.53         9
         R64       0.64      0.78      0.70        18
         S16       0.25      0.20      0.22         5

    accuracy                           0.54        39
   macro avg       0.25      0.27      0.25        39
weighted avg       0.47      0.54      0.50        39

Accuracy on the test set (top 5 features): 0.5385
```

Logistic Regression:

1) The dataset is split into 2 parts: the “20% dataset” (for testing) and “80% dataset” (for feature importance calculating, parameter tuning, and model training).
2) We then ran feature importance calculations on all columns in order to exclude columns with negligible importance.



3) Grid search performed on 80% training data in order to find best hyper parameter settings based on “accuracy” metric.

Best parameters: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}

4) The remaining “20% dataset” will be used to test and evaluate the model by calculating accuracy, precision, recall, and F1-score.

```
Classification Report:
              precision    recall  f1-score   support

         2ND       0.00      0.00      0.00         1
   Champions       0.00      0.00      0.00         1
          E8       0.00      0.00      0.00         1
          F4       0.00      0.00      0.00         0
         R32       0.44      0.36      0.40        11
         R64       0.67      0.89      0.76        18
         S16       0.40      0.29      0.33         7

    accuracy                           0.56        39
   macro avg       0.22      0.22      0.21        39
weighted avg       0.50      0.56      0.52        39
```

Random Forest:

1) The dataset is split into 2 parts: the “20% dataset” (for testing) and “80% dataset” (for feature importance calculating, parameter tuning, and model training).
2) We decided to keep all common features within the training/test/validation data since there were no issues with runtime or performance cost.
3) Grid search performed on 80% training data in order to find best hyper parameter settings based on “accuracy” metric.

```
Best parameters: {'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}
```

4) The remaining “20% dataset” will be used to test and evaluate the model by calculating accuracy, precision, recall, and F1-score.

Model Classification Report:

	precision	recall	f1-score	support
R64	0.58	0.75	0.65	20
R32	0.11	0.14	0.12	7
S16	0.33	0.40	0.36	5
E8	0.00	0.00	0.00	5
F4	0.00	0.00	0.00	1
2ND	0.00	0.00	0.00	0
Champions	0.00	0.00	0.00	1
micro avg	0.44	0.46	0.45	39
macro avg	0.15	0.18	0.16	39
weighted avg	0.36	0.46	0.40	39

3. BUILD MODEL

3.1. Parameter settings

1) KNN: The main parameter is the number of neighbors, k. A range of values from 2 -29 (the recommended k value range) were each used and evaluated to find which was the best k. This was by finding the knn test score for each k and choosing the k that resulted in the highest score. When the data was split with 20% set for the testing dataset, k = 2,3 and 20 had the highest test score and were used as the k value. When the split was set at 30%, the best k was suggested to be 9, 20, 28.

2) SVM: Based on feature importance calculation, we will be using the top 9 columns in terms of feature importance to train on. We are using standard normalization for numerical values before modeling the data using SVM. Based on grid search results, we will use the Linear kernel. Additionally, we will also use the following hyperparameter settings: {C': 1, 'coef0': 0, 'degree': 2, 'gamma': scale}. For this model we removed R68 teams.

3) Logistic Regression: Similar to the SVM model, for the logistic regression model we will be using the top 9 columns as the 10th column we see a large drop off in importance. We also used a grid search for the hyperparameters for the logistic regression and ended up using {'C': 1, 'penalty': 'l2', 'solver': 'saga'}. For this model we removed R68 teams.

4) Random Forest: Using the default 100 trees and no depth limit. Features are scaled using 'StandardScaler' to normalize the data, ensuring that all features contribute equally to the model's training process. For this model we removed R68 teams.

3.2. Models

At first Purdue was said to be out in the first round for all our tested models, which made us realize we needed to make some adjustments to our models as well as go back to the data preparation stage. First we realized the model did not know the rules that only a certain number of teams will get in the first round, etc. So we had to create constraints in the model that indicated how many teams would get eliminated in each

round. We also realized that the four teams that made it a total of 72 instead of the traditional 68 teams made a negative impact on our model. So by removing those teams from our data and implementing the constraints our predictions for Purdue's placement jumped to Sweet 16 and Elite 8. After fine tuning the hyperparameters, Purdue's placement was consistently the Final 4.

We run the following models from Scikit-Learn:

1) KNN

The first models had a testing split of 0.2. There were 3 k's that had the top test scores that were used.

k = 2: purdue S16, acc: .48

```
Model accuracy: 0.48780487804878005
```

	TeamName	PredictedStage
0	Akron	Champions
1	Alabama	2ND
2	Florida Atlantic	F4
3	Illinois	F4
4	Auburn	E8
...	...	...
59	Colorado	R64
60	Colgate	R64
61	Vermont	R64
62	Nevada	R64
63	Virginia	R64

64 rows × 2 columns

```
   TeamName PredictedStage
12   Purdue            S16
```

k =3 purdue S16, acc: 0.46

```
Model accuracy: 0.4634146341463415
```

	TeamName	PredictedStage
0	Akron	Champions
1	Alabama	2ND
2	Florida Atlantic	F4
3	Gonzaga	F4
4	Auburn	E8
...	...	...
59	Colorado	R64
60	Colgate	R64
61	Wagner	R64
62	South Dakota St.	R64
63	Yale	R64

64 rows × 2 columns

```
   TeamName PredictedStage
10   Purdue            S16
```

k=20: purdue S16, acc:0.39

```
Model accuracy: 0.3902439024390244
```

	TeamName	PredictedStage
0	Connecticut	Champions
1	Auburn	2ND
2	Houston	F4
3	James Madison	F4
4	Iowa St.	E8
...	...	...
59	McNeese St.	R64
60	Charleston	R64
61	Northwestern	R64
62	UAB	R64
63	Colorado	R64

64 rows × 2 columns

```
   TeamName PredictedStage
15   Purdue            S16
```

The first models had a testing split of 0.3. There were 2 k's that had the top test scores that were used.

k=25,Purdue F4 acc = 0.48

```
Model accuracy: 0.4838709677419355
```

	TeamName	PredictedStage
0	Connecticut	Champions
1	Alabama	2ND
2	Purdue	F4
3	Baylor	F4
4	Arizona	E8
...	...	...
59	Colgate	R64
60	Akron	R64
61	Oregon	R64
62	Wagner	R64
63	Illinois	R64

64 rows × 2 columns

```
  TeamName PredictedStage
2   Purdue              F4
```

k=28,Purdue F4 acc = 0.48

```
Model accuracy: 0.4838709677419355
```

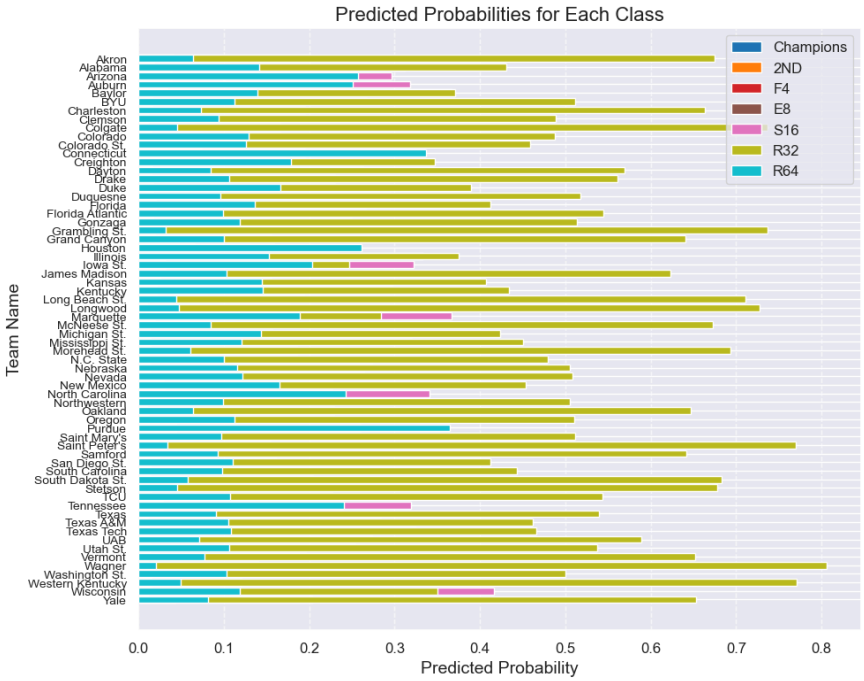
	TeamName	PredictedStage
0	Connecticut	Champions
1	Alabama	2ND
2	Auburn	F4
3	Purdue	F4
4	Arizona	E8
...	...	...
59	Colgate	R64
60	Morehead St.	R64
61	Duquesne	R64
62	Longwood	R64
63	Gonzaga	R64

64 rows × 2 columns

```
  TeamName PredictedStage
3   Purdue              F4
```

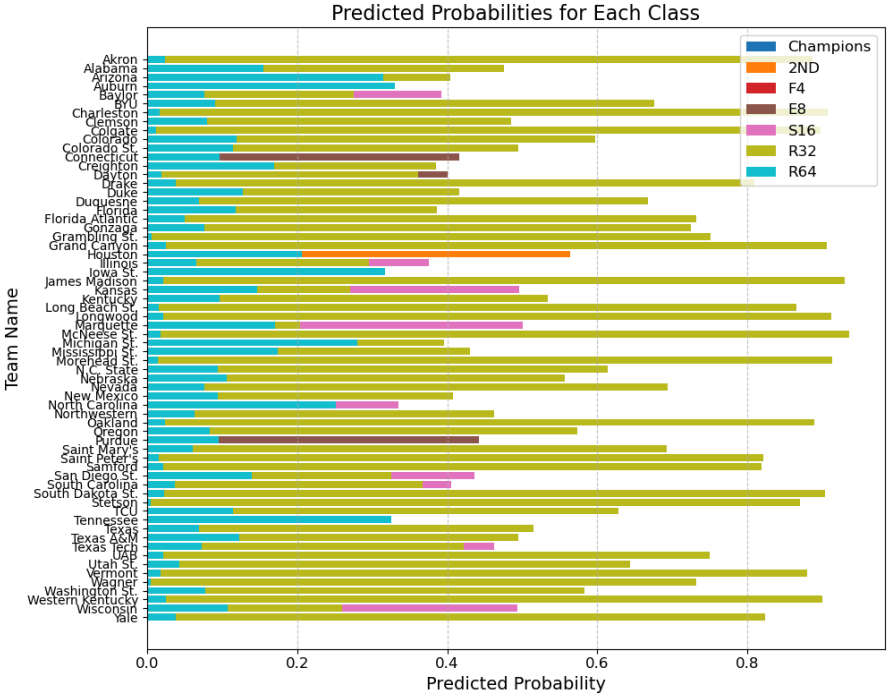
2) SVM – SVC

Probabilities for each team and each classification:



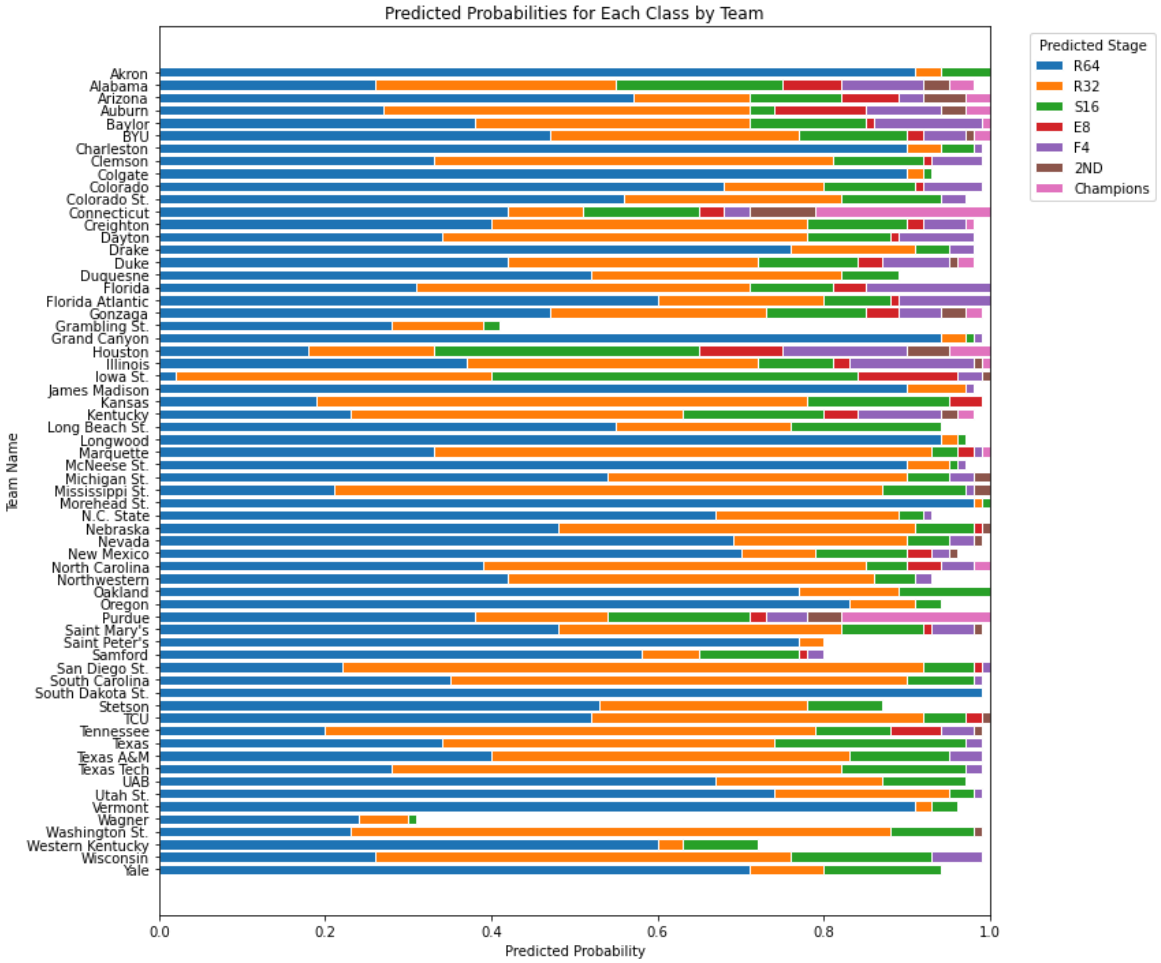
3) Logistic Regression – linear_model.LogisticRegression

Probabilities for each team and each classification:

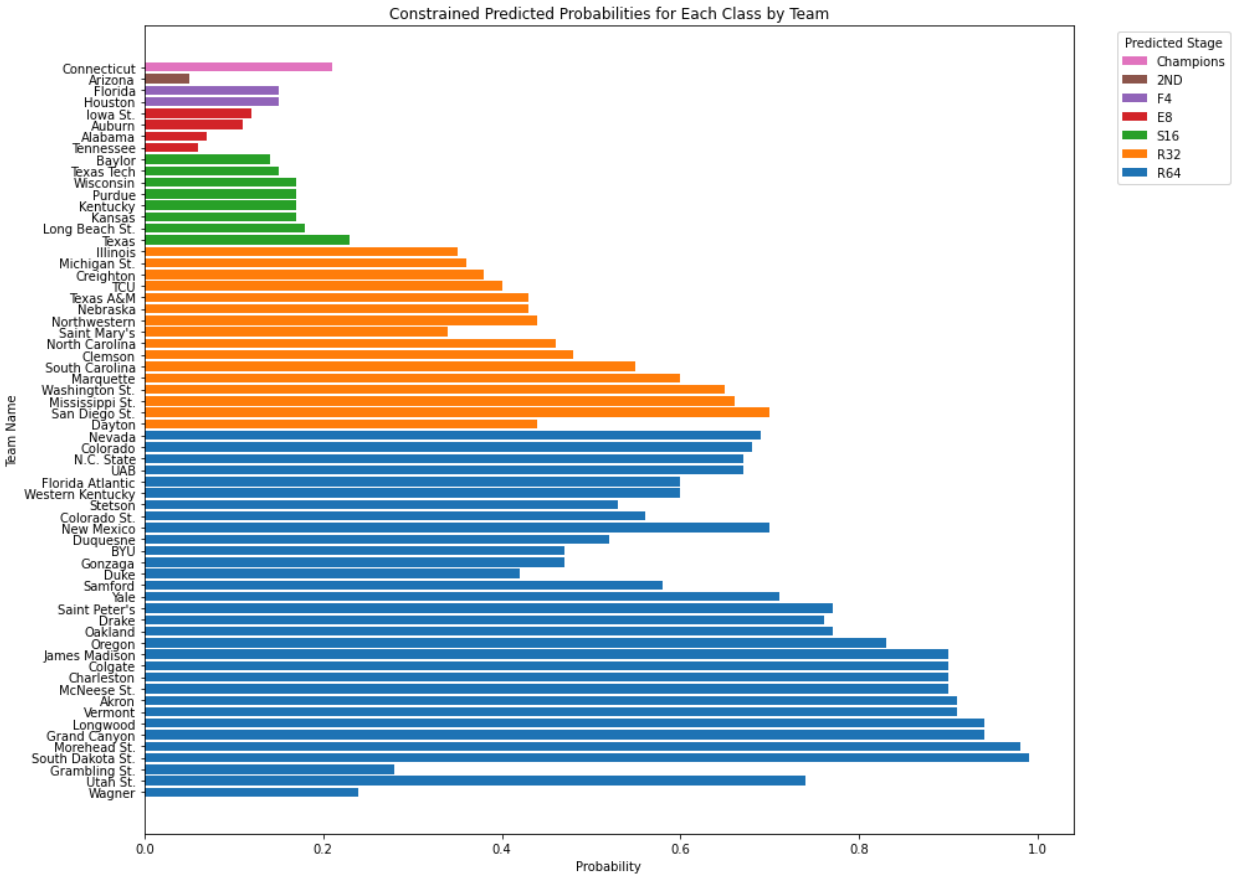


4) Random Forest – ensemble.RandomForestClassifier

Probabilities for each team and each classification:



Final team predictions after choosing the team with the highest predicted stage probability for each classification:



One strategy used for all of our models for post-processing the predictions was Platt scaling. Given that the predictions must follow particular quotas for each classification, we opted to use platt scaling to adjust our predictions such that each classification of 'POSTSEASON' would reach its necessary number of values.

3.3. Model description

For all the models we introduced a rule of constraints. We were noticing that more teams were being predicted to be eliminated in a round than were allowed based on the championships rules. This resulted in the majority of the teams being assigned to getting eliminated in the first round, including Purdue. To account for this we created a constraint rule, which included a function that applied a dictionary of each round's name and the number of teams playing in that round. This constraint rule produced more accurate results and did a better job of simulating the rounds. Afterwards Purdue's overall performance (as well as other teams) improved to getting eliminated in the Sweet 16 or higher.

No opaque models were used in this project.

Limitation: By analyzing the results, it would sometimes report UCONN being the champion and Alabama being the 2nd place winner. However this is not possible as UCONN and Alabama play each other in the final 4 so only one of the teams could be the champion or 2nd place. This is a limitation and difficult to fix because it is unknown who is going to play each other, and if the match up is even possible based on the schedule.

We are running four different models:

1) KNN:
- Simple model, that has a strong accuracy.
- Robust and limited assumptions about the data meaning it can be used on non-linear data with missing values and is useful if the data is not as clean in the future.
- Limitation: It can be computationally expensive if the data is large (or if it is to get larger in the future), however, as of now we have a relatively small dataset.
- Selected parameters: {‘k’:25 or 28, ‘metric’ = 'minkowski', p=2} with a split of 0.3 for the more accurate results.
    - Both k values have the same test score.
    - The metric and p are the default values.

2) SVM:
- Effective in high dimensional spaces, even if the number of dimensions is greater than the number of samples and is memory efficient
- One limitation is that SVM is very sensitive to noise and the unpredictable nature of upsets in march madness creates a potential weakness here.
- Selected parameters: {‘C': 1, 'coef0': 0, 'degree': 2, 'gamma': scale, ‘kernel’:linear}

3) Logistic regression:
- Uses a multinomial approach because of the multiclass problem.
- It models the probability of an event occurring as a function of the predictor variables.
- It is a simple and interpretable model, making it easy to understand the relationship between the features and the target variable.
- Logistic regression does not assume a linear relationship between the predictor variables and the target variable, making it suitable for scenarios where the relationship is non-linear.
- Selected parameters: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}

4) Random Forest:
- An ensemble method that works well at classification tasks with low or high number of features and can handle non-linear relationships
- Can become computationally intensive with large number of features (not an issue with our dataset)
- Selected parameters: {'max_depth': None, 'n_estimators': 100}

Chosen (non-default) parameters:

1) SVM
- {C': 1, 'coef0': 0, 'degree': 2, 'gamma': scale, ‘kernel’:linear}

2) Logistic regression (linear_model.LogisticRegression)

- {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}

3) Random forest (ensemble.RandomForestClassifier)

- {'n_estimators': 100}

SVM constructs a hyperplane boundary and maximizes the distance of any class' nearest data point (margin). Based on the magnitude of the weight for each feature, the top 5 features are: FUTURE_ODDS, W, SEED, AdjEM, and DE

Multinomial logistic regression is like a voting system for multiple categories. It analyzes data to estimate the odds of something belonging to each category, considering various factors all at once. In our case it uses the team's attributes to estimate which round they will get eliminated in March Madness.

Random forest is an ensemble learning method, constructing multiple decision trees at training time. The output of random forest is the selected class by most of the trees. Random forest corrects the overfitting issue of decision trees.

4. ASSESS MODEL

We used a Train and Test set as our testing strategy to evaluate our predictions. Our test set was "20%" as that is the advised initial parameter value. We use accuracy and weighted macro average (to account for class imbalance) for precision, recall, and F1-score to evaluate the models. The evaluation metrics for initial runs for all three algorithms are given below:

	Evaluation metrics on the "20%" test set (Initial Parameters)			
	Accuracy	**Precision**	**Recall**	**F1-score**
Logistic Regression	.5641	.5	.56	.52
SVM	0.4615	0.35	0.46	0.39
KNN	0.43	0.33	0.43	0.37
Random Forest	0.4146	0.16	0.18	0.17

Logistic Regression:

```
Classification Report:
              precision    recall  f1-score   support

         2ND       0.00      0.00      0.00         1
   Champions       0.00      0.00      0.00         1
          E8       0.00      0.00      0.00         1
          F4       0.00      0.00      0.00         0
         R32       0.44      0.36      0.40        11
         R64       0.67      0.89      0.76        18
         S16       0.40      0.29      0.33         7

    accuracy                           0.56        39
   macro avg       0.22      0.22      0.21        39
weighted avg       0.50      0.56      0.52        39
```

SVM:

```
Classification Report:
              precision    recall  f1-score   support

         2ND       0.00      0.00      0.00         1
   Champions       0.00      0.00      0.00         1
          E8       0.00      0.00      0.00         4
          F4       1.00      1.00      1.00         1
         R32       0.20      0.12      0.15         8
         R64       0.50      0.79      0.61        19
         S16       0.33      0.20      0.25         5

    accuracy                           0.46        39
   macro avg       0.29      0.30      0.29        39
weighted avg       0.35      0.46      0.39        39

Accuracy on the test set (top 5 features): 0.4615
```

KNN:

Accuracy Score : 0.4305555555555556
Report :

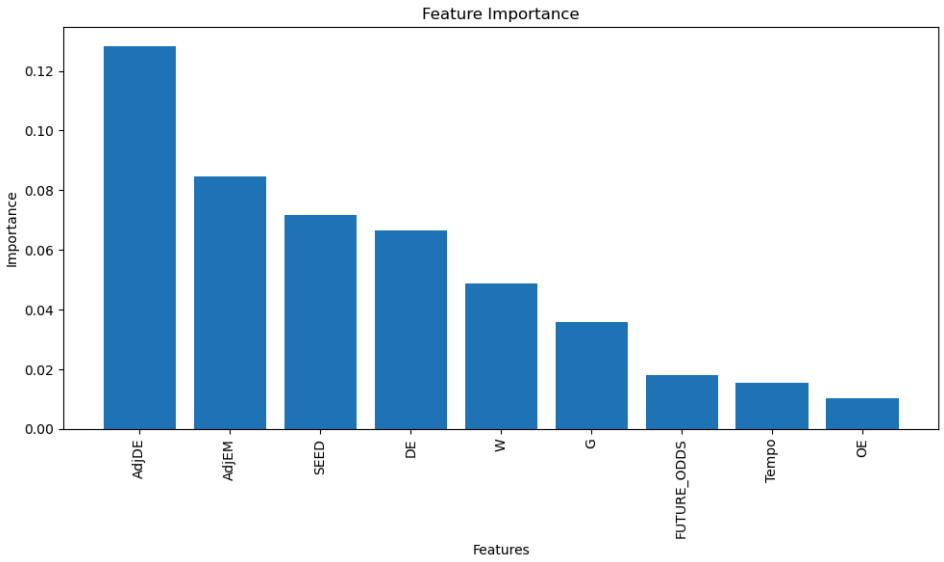
	precision	recall	f1-score	support
2ND	0.00	0.00	0.00	1
E8	0.00	0.00	0.00	4
F4	0.25	1.00	0.40	1
R32	0.32	0.33	0.32	18
R64	0.55	0.73	0.62	33
R68	0.00	0.00	0.00	3
S16	0.00	0.00	0.00	12
accuracy			0.43	72
macro avg	0.16	0.29	0.19	72
weighted avg	0.33	0.43	0.37	72

Random Forest:

Classification Report:

	precision	recall	f1-score	support
R64	0.54	0.70	0.61	20
R32	0.09	0.14	0.11	7
S16	0.50	0.40	0.44	5
E8	0.00	0.00	0.00	5
F4	0.00	0.00	0.00	1
2ND	0.00	0.00	0.00	0
Champions	0.00	0.00	0.00	1
micro avg	0.41	0.44	0.43	39
macro avg	0.16	0.18	0.17	39
weighted avg	0.36	0.44	0.39	39

From the table, it seems that the logistic regression model performs the best without any hyperparameter tuning. Without hyperparameter tuning, SVM, KNN, and Random Forest appear to get a very low accuracy. Keep in mind that these numbers are also without Platt scaling for SVM. Hyperparameter tuning finds the optimal values for the parameters which should increase the accuracy of the results which is why we decided to use this method.



The top 5 features for the best model (Logistic Regression) are:

1) AdjDE
2) AdjEM
3) SEED
4) DE
5) W

Based on the predictions for SVM, it is clear that it does not yet produce plausible predictions. Given that no feature engineering or post prediction adjusting have been performed, this result should be expected. SVM is particularly sensitive to noise, and as such, we expected to have low accuracy rates for the initial prediction.

KNN improved accuracy when the k had the highest test_score associated with it. This score was dependent on the train/test split. Originally the split was 0.2, however the 0.3 split made more realistic predictions. This 30% split was recommended by empirical sources.

Random Forest quickly provides an explainable model without any extensive hyperparameter tuning. However, after analyzing the evaluation metrics compared to all other models, it is clear to see that the Random Forest model does not yet produce reliable predictions. This may be due to insufficient model complexity or lack of feature engineering.

There are logical rules to all the models that involve constraints to the number of teams allowed to be in each round (a total of 8). This follows common sense and the rules of the tournament and increases the accuracy of our results.

In terms of business, the results for all models are around 0.5 meaning it is performing around random. However, with more data, more fine tuning of the parameters, and improvements of the model will increase these scores. It is important to note that it is predicting Purdue's performance consistently which

demonstrates repeatability and hopefully reliability. It is also predicting Purdue and the majority of the final four correctly as of now. This implies plausibility and even accuracy however we know based on the lower scores this is not the case. Vegas can use these predictions to calculate their odds and users can use it to help create a more accurate bracket. However, these evaluation scores are low, meaning it is risky to make financial or business decisions based on this model in its current state. If this is improved in the future, then this model could be deployed for both personal and business use. Especially if the evaluation scores increase to be reliable then teams who are predicted to advance to the top rounds can start creating merch, selling tickets ahead of time etc., and lower teams and focus on how they can improve.

As a reminder our data mining goal was to predict Purdue's chances of winning the 2024 NCAA Tournament, given previous teams' performances with similar regular season statistics. We required that accuracy, precision, recall, and F1-score be greater than 0.80, which was unfortunately not achieved during the set timeline for this project. However, our success criteria was defined by correctly predicting the number of games Purdue will play in the tournament with an error of 1 round, which was accomplished.

In conclusion, the ranking is: Logistic Regression > SVM > KNN > Random Forest

4.2. Revised parameter settings

We perform a grid search on some of the algorithms to find the best parameters.

1) SVM:

- The regularization parameter C should be decided from the set [0.01, 0.1, 1, 10, 100].
- Kernel should be decided on the set ['linear', 'rbf', 'poly', 'sigmoid']
- Kernel coefficient should be decided on the set ['scale', 'auto', 0.1, 1, 10]
- Degree should be decided on the set [2, 3, 4]
- Constant term for kernel should be decided on the set [0, 0.1, 1, 10]

2) Logistic Regression:

- The parameter for inverse of regularization strength C should be chosen from {0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100}.
- The parameter for penalty should be decided on the set {'l1', 'l2'}
- The parameter for solver should be decided on the set {'liblinear', 'saga'}

3) Random Forest:

- The parameter for the number of estimators should be chosen from the set: {50, 100, 200}.
- The parameter for the maximum depth of the tree should be chosen from the set: {None, 10, 20, 30}.
- The parameter for the minimum number of samples required to split an internal node should be chosen from the set: {2, 5, 10}.
- The parameter for the minimum number of samples required to be at a leaf node should be chosen from the set: {1, 2, 4}.

- The parameter for the number of features to consider when looking for the best split should be chosen from the set: {‘auto’, ‘sqrt’}.

For the KNN model, 27 different k values were chosen to provide a range of possible values. For each k, the train and test scores were produced and a k of 25 and 28 had the highest test score for the classifier was 0.51 (when the test train split was 0.3). When the split was 20%, the k values of 2, 3, and 20 had the highest value of 0.51. The metric and p-value were unchanged as the default was proven to be the most accurate.

The best parameters found by using the grid search strategy on some of our algorithms is as follows:

1) SVM: {'C': 1, 'coef0': 0, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'}
2) Logistic Regression: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
3) Random Forest: {'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}

The evaluation metrics for best parameter runs for all four algorithms are given below. Weighted macro average (to account for class imbalance) is used for precision, recall, and F1-score.

	Evaluation metrics on the “20%” test set (Best parameters)			
	Accuracy	**Precision**	**Recall**	**F1-score**
Logistic regression	0.6667	0.70	0.67	0.68
SVM	0.5385	0.47	0.54	0.50
KNN	0.4839	0.39	0.48	0.43
Random Forest	0.439	0.15	0.18	0.16

Logistic Regression after grid search:

```
Classification Report:
              precision    recall  f1-score   support

         2ND       0.00      0.00      0.00         1
   Champions       0.00      0.00      0.00         1
          E8       0.00      0.00      0.00         3
          F4       0.00      0.00      0.00         0
         R32       0.43      0.50      0.46         6
         R64       0.92      0.85      0.88        27
         S16       0.00      0.00      0.00         1

    accuracy                           0.67        39
   macro avg       0.19      0.19      0.19        39
weighted avg       0.70      0.67      0.68        39

Accuracy on the test set (top 9 features): 0.6667
```

SVM after grid search:

```
Classification Report:
              precision    recall  f1-score   support

         2ND       0.00      0.00      0.00         2
   Champions       0.00      0.00      0.00         1
          E8       0.00      0.00      0.00         1
          F4       0.33      0.33      0.33         3
         R32       0.50      0.56      0.53         9
         R64       0.64      0.78      0.70        18
         S16       0.25      0.20      0.22         5

    accuracy                           0.54        39
   macro avg       0.25      0.27      0.25        39
weighted avg       0.47      0.54      0.50        39

Accuracy on the test set (top 5 features): 0.5385
```

KNN after grid search:

Classification Report:

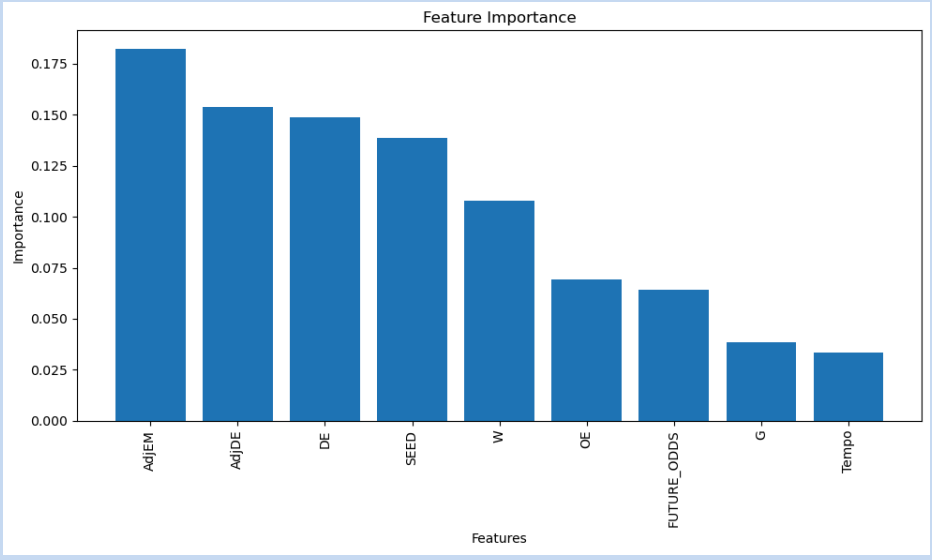
	precision	recall	f1-score	support
Champions	0.00	0.00	0.00	1
2ND	0.00	0.00	0.00	2
F4	0.00	0.00	0.00	5
E8	0.00	0.00	0.00	1
S16	0.17	0.36	0.24	11
R32	0.67	0.79	0.72	33
R64	0.00	0.00	0.00	3
R68	0.00	0.00	0.00	6
accuracy			0.48	62
macro avg	0.11	0.14	0.12	62
weighted avg	0.39	0.48	0.43	62

Random Forest after grid search:

Model Classification Report:

	precision	recall	f1-score	support
R64	0.58	0.75	0.65	20
R32	0.11	0.14	0.12	7
S16	0.33	0.40	0.36	5
E8	0.00	0.00	0.00	5
F4	0.00	0.00	0.00	1
2ND	0.00	0.00	0.00	0
Champions	0.00	0.00	0.00	1
micro avg	0.44	0.46	0.45	39
macro avg	0.15	0.18	0.16	39
weighted avg	0.36	0.46	0.40	39

In conclusion, the ranking is: Logistic regression > SVM > KNN > Random Forest



The top 5 features for the best model (Multinomial Logistic Regression) are:

1) AdjEM
2) AdjDE
3) DE
4) SEED
5) W

These are the same top 5 features as before, however, they are in a different order with the hyperparameter tuning after the grid search. We also see a similar drop off in feature importance after the 5th feature.