Author: Saravanan Natarajan
Indiana University
sanata@iu.edu

Author: Senthil Palani
Indiana University
spalani@iu.edu

Title: Deep learning – NACC Basketball March Madness Analysis & Prediction

Keywords: Feature Engineering, Feature generation, feature *importance,* Feature Selection, KFold, data Standardization, Machine Learning, Deep learning, pipeline, estimators**,** grid search, Bagging, ensemble, confusion matrix, Statistical Significance Test, Models Accuracy plot.

Abstract:

The classic problem of sport informatics is baseball win prediction and it has been explored with conventional machine learning and deep learning models. In this paper, we present a comprehensive Feature Analysis, Machine learning, Deep learning models and comparison of various model performances. The feature analysis is very important for users to efficiently solve any machine and deep learning problems. A comprehensive feature analysis and interpretation of sports informatics basketball dataset is carried out in this paper. Specifically, we classify the feature analysis into four categories: Feature engineering, Feature generation, feature importance and features selection to make it mode compressive analysis. The final features are algorithmically and visually evaluated and extracted carefully and feeding to various training and prediction models. Finally, various model accuracy is compared and plotted to identify the best model for the given dataset. Possible future research opportunities are also explored and discussed.

Introduction:

The goal of our project is to predict the outcomes of college basketball games during a season based on key performance variables for each team. There are ~5500 Men's Division 1 basketball games each year. Our intent is to train our model on the first 80% of games played using 16 key performance variables recorded during the game by each team, and compare to the outcome of the game. We will then predict the winners of the remaining 20% of games.

The interesting part of this project is determining how to deal with the 'test data' for the final 20% of the games. We cannot simply use the performance variables captured during the 'test' games to predict the winner, because in reality you would not know these parameters until after the game is played (steals, shot percentage, total points, etc.). Instead we will have to 'predict' each team expected game performance features based on their previous history, and then run the model using those features to create the ultimate target: the game winner.

In this project, we developed and assessed machine learning and deep learning classifier models to tackle the basketball sports informatics dataset. First, we created a machine learning classification model that attempted to address the basketball prediction. For this model, we compared SVC, LogisticRegression, SGDClassifier, GaussianNB, RandomForestClassifier, KNeighborsClassifier, DecisionTreeClassifier. Second, we used a deep learning neural network models fully connected neural network models. This neural network model could serve as the basis for a future analysis and study in this basketball sports informatic space.

Overall process

Dataset

We are utilizing data from a Kaggle competition based on predicting the 2018 NCAA Men's College Basketball Tourney results. We will re-purpose the data for our study. The key data set that we will utilize will include the 16 key performance indicators for each team during every game of the season. This data set contains every game going back to 2003. We will treat each season as a separate study. Our intent is to predict the 2017-2018 season, but our stretch goal is to predict additional years and see whether the optimal method is the same or different across years.

Data Pre-processing

(i) Feature Engineering

The input baseball dataset comprises features, which are usually in the form of structured columns. Algorithms require features with some specific characteristic to work properly. Here, the need for **feature engineering** arises. The feature engineering efforts mainly have two goals. The first goal is preparing the proper input dataset, compatible with the machine learning algorithm requirements. Secondly, Improving the performance of machine learning models. The features you use influence more than everything else the result. No algorithm alone, to my knowledge, can supplement the information gain given by correct **feature engineering**.

First, we removed the team performance variables that directly impact the result of the game. Any of the fields That indicated points scored was removed. Instead we created new fields that showed the shooting percentage. Shooting percentage is indicative of performance without giving a 1-to-1 ratio to winning. After completing many logistic regressions runs using Lasso and Ridge, we evaluated the most impactful coefficients We found that 3-point shooting, defensive rebounds and turnovers seemed to always be the most important. To capitalize on this, we created many new features that compared the ratio of these variables between the two competing teams. In our initial runs, we found that these variables increased the overall accuracy of the model by 5% This looks like a good area to look at in future runs.

(ii) Feature Generation

Feature generation is the process of creating new features from one or multiple existing features, potentially for using in statistical analysis. This process adds new information to be accessible during the model construction and therefore hopefully result in more accurate model.

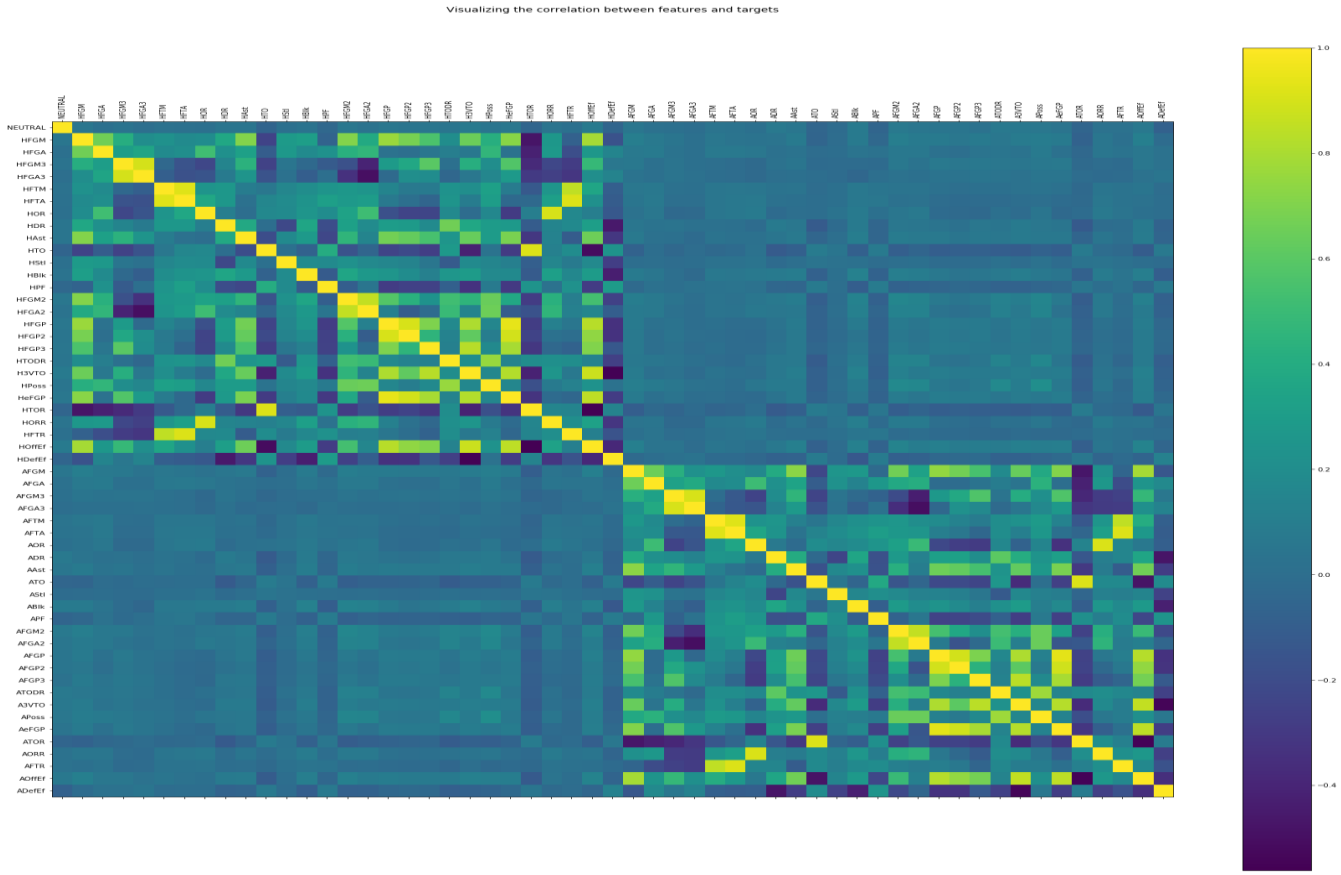
Example:

```
HomeAwayResults['HFGM2']=HomeAwayResults['HFGM'] - HomeAwayResults['HFGM3'] #HOME 2 POINT FIELD GOALS MADE
HomeAwayResults['AFGM2']=HomeAwayResults['AFGM'] - HomeAwayResults['AFGM3'] #AWAY 2 POINT FILED GOALS MADE
HomeAwayResults['HFGA2']=HomeAwayResults['HFGA'] - HomeAwayResults['HFGA3'] #HOME 2 POINT FIELD GOALS ATTEMPTED
HomeAwayResults['AFGA2']=HomeAwayResults['AFGA'] - HomeAwayResults['AFGA3'] #AWAY 2 POINT FIELD GOALS ATTEMPTED

# Number of Possessions
HomeAwayResults['HPoss']= (HomeAwayResults['HFGA'] - HomeAwayResults['HOR'] + HomeAwayResults['HTO'] + (0.44 * HomeAwayResults['HFTA'])) # HOME Number of Possessions
HomeAwayResults['APoss']= (HomeAwayResults['AFGA'] - HomeAwayResults['AOR'] + HomeAwayResults['ATO'] + (0.44 * HomeAwayResults['AFTA'])) # AWAY Number of Possessions
```

(iii) **Further Feature Analysis:**

Feature correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other. Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features. Removal of different features from the dataset will have different effects on the p-value for the dataset. We can remove different features and measure the p-value in each case. These measured p-values can be used to decide whether to keep a feature or not.



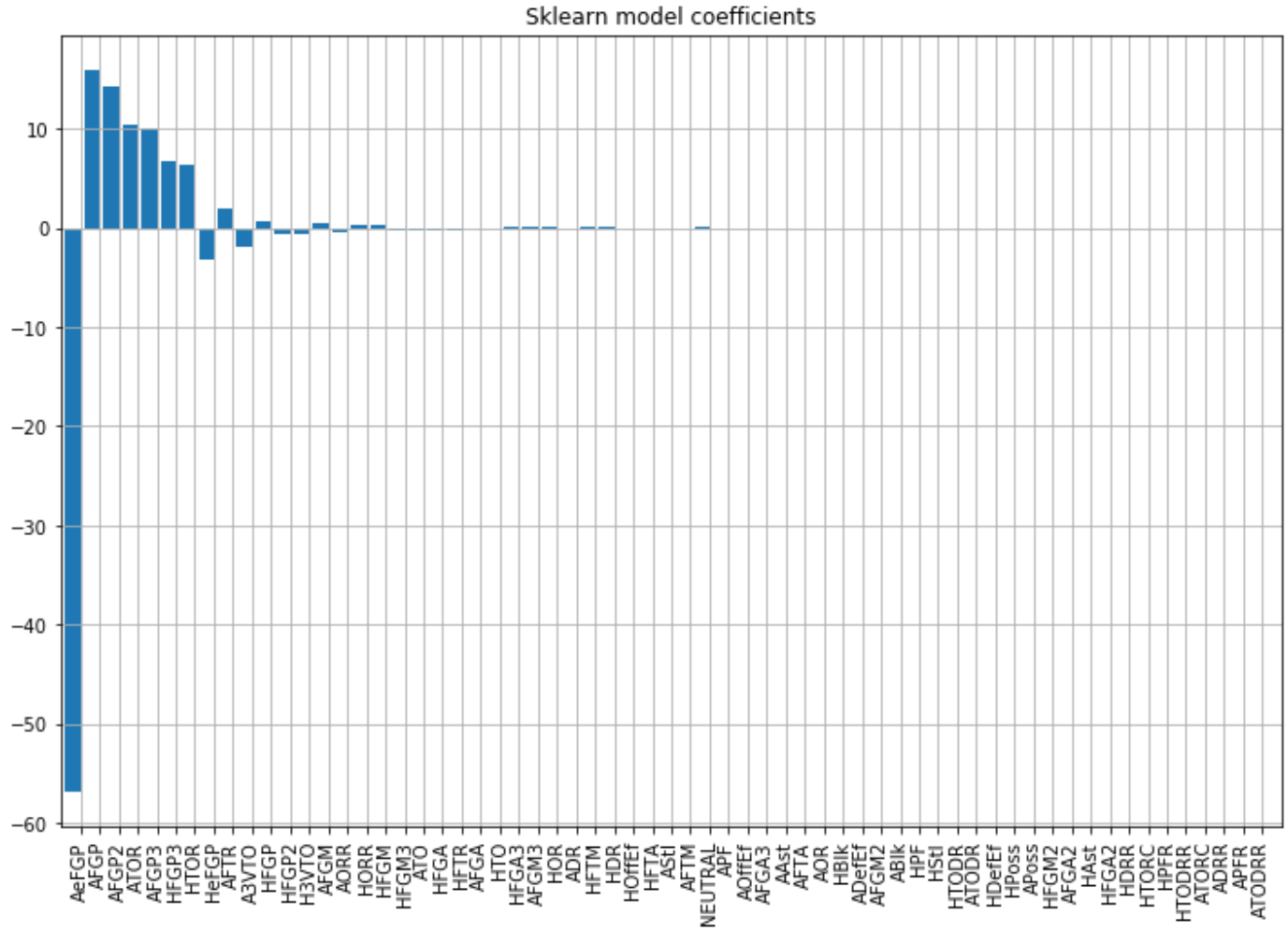
Feature Correlation Plotting: Initial all available features

(iv) Recursive Feature Elimination

The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute. You can learn more about the RFE class in the scikit-learn documentation. RFE with the logistic regression algorithm to select the top features. The choice of algorithm does not matter too much as long as it is skillful and consistent.

Analyzing Feature with high correlation [corr greater that 0.75]

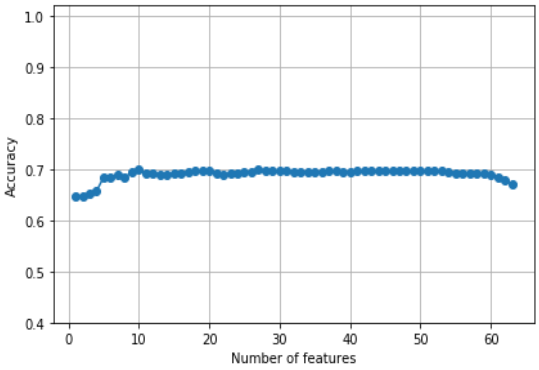
Feature analysis: Regression model - Scoring coefficients



(v) Feature Importance

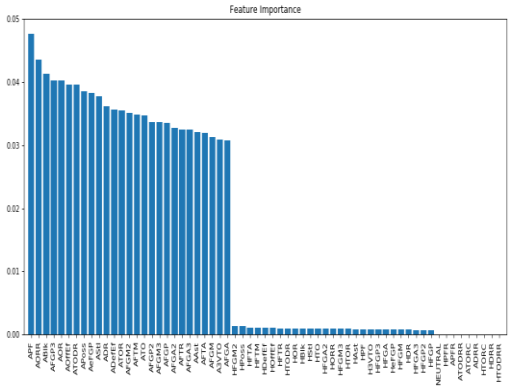
Feature Importance Using Random Forest

Random Forests are often used for feature selection in a data science workflow. The reason is because the tree-based strategies used by random forests naturally ranks by how well they improve the purity of the node. This mean decrease in impurity over all trees (called gini impurity). Nodes with the greatest decrease in impurity happen at the start of the trees, while notes with the least decrease in impurity occur at the end of trees. Thus, by pruning trees below a particular node, we can create a subset of the most important features.



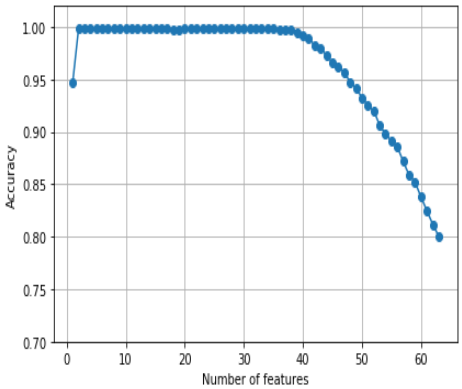
(v) Feature Selection

Often in data science we have hundreds or even millions of features and we want a way to create a model that only includes the most important features. This has three benefits. First, we make our model simpler to interpret. Second, we can reduce the variance of the model, and therefore overfitting. Finally, we can reduce the computational cost (and time) of training a model. The process of identifying only the most relevant features is called "feature selection."



Sequential Backward Selection:

Sequential feature selection algorithms are a family of greedy search algorithms that are used to reduce an initial *d*-dimensional feature space to a *k*-dimensional feature subspace where $k < d$. The motivation behind feature selection algorithms is to automatically select a subset of features that is most relevant to the problem. The goal of feature selection is two-fold: We want to improve the computational efficiency and reduce the generalization error of the model by removing irrelevant features or noise.



Feature Selection: Sequential feature selection algorithms: LogisticRegression:

A wrapper approach such as sequential feature selection is especially useful if embedded feature selection. In a nutshell, SFAs remove or add one feature at the time based on the classifier performance until a feature subset of the desired size *k* is reached.

Classification using Deep Learning

The results for each deep learning classifier can be seen in Table below. We have tried different variation of DL models to extract the best model out of list of experimented models. We used vanilla neural network model, grid search implementation, estimators and pipeline techniques to derive the accuracy for various machine learning models. As the resulting table shows, we got maximum accuracy using the following Deep learning fully connected neural model with root mean square optimizer.

All of our data was numerical, so our use of a pipeline was straightforward we used a MinMaxScaler to standardize the data, and used a classifier for the method. We also used a parameter grid for the penalty and C values, and then put it into a grid. We have run this dozens of times in order to evaluate our features, but in this first phase we have not experimented with other types of methods -- we will explore those in future runs.

We tried following different models and calculated accuracy for each model using various parameters/pipeline approaches etc.

```
runResultsDL
```

	Model_Description	Bagging	accuracy_score
0	DEEP LEARNING BASE model	Estimator	0.601000
1	DEEP LEARNING BASE model Without standardization	No	0.624000
2	DEEP LEARNING BASE model - GRID SEARCH BEST PARAM IMPL	No	0.525333
3	DEEP LEARNING BASE model - ROOT MEAN SQUARE OPTIMIZER	No	0.624000
4	DEEP LEARNING BASE model - Stochastic gradient descent (SGD) OPTIMIZER	No	0.376000

Best machine learning model test accuracy of approximately 63% [green colored highlighted box]. The worse model test accuracy was from the stochastic gradient descent classifier [red colored highlighted box].

Classification using Conventional Machine Learning

The results for each machine learning classifier can be seen in Table below. We have tried different variation of models to extract the best model. We used vanilla model, grid search implementation, estimators, pipeline and bagging techniques to derive the accuracy for various machine learning models. As the resulting table shows, we got maximum accuracy using the following Grid Search DecisionTreeClassifier Estimator with Standard Scaler standardization.

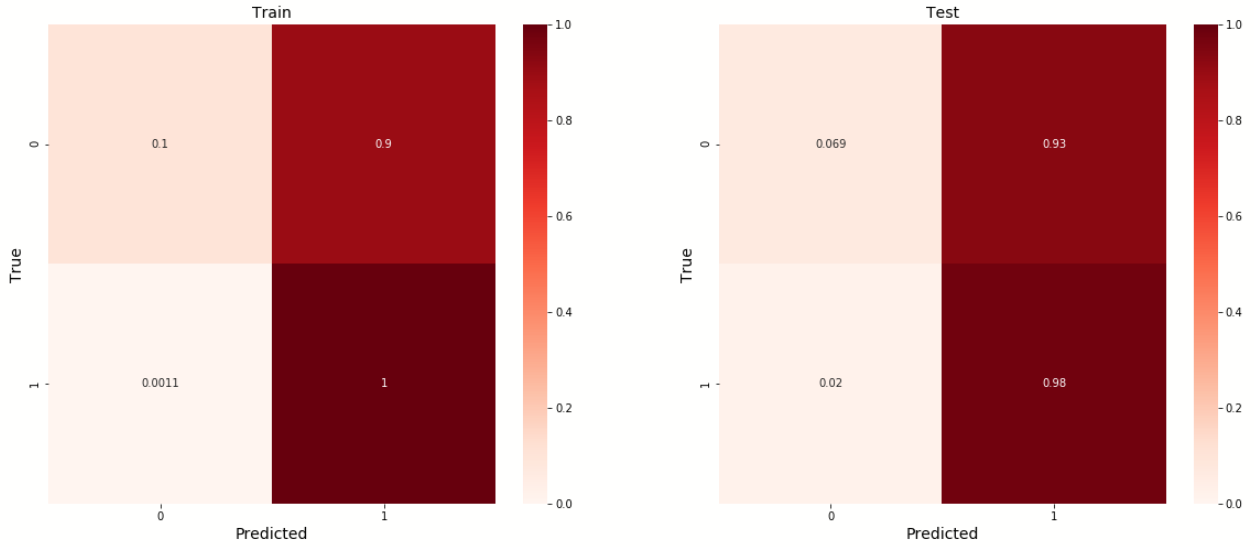
We tried following different models and calculated accuracy for each model using various parameters/pipeline process/Bagging approaches etc.

1. SVC
2. LogisticRegression
3. SGDClassifier
4. GaussianNB
5. RandomForestClassifier
6. KNeighborsClassifier
7. DecisionTreeClassifier

	Model_Description	Bagging	accuracy_score
0	LogisticRegression/RandomForestClassifier/GaussianNB	Estimator	0.601000
1	DecisionTreeClassifier	No	0.549333
2	BaggingClassifier - DecisionTreeClassifier	Yes	0.610667
3	LogisticRegression	No	0.584889
4	BaggingClassifier - LogisticRegression	Yes	0.591111
5	KNeighborsClassifier	No	0.590222
6	BaggingClassifier - KNeighborsClassifier	Yes	0.627556
7	RandomForestClassifier-StandardScaler	No	0.550222
8	BaggingClassifier - RandomForest-StandardScaler	Yes	0.617778
9	RandomForestClassifier-MinMaxScaler	No	0.550222
10	BaggingClassifier - RandomForest-MinMaxScaler	Yes	0.618667
11	SVC-StdScaler	No	0.604444
12	BaggingClassifier - SVC-StdMaxScaler	Yes	0.624000
13	SVC-MinMaxScaler	No	0.604444
14	BaggingClassifier - SVC-MinMaxScaler	Yes	0.624000
15	SGDClassifier-StandardScaler	No	0.517333
16	BaggingClassifier - SGDClassifier-StandardScaler	Yes	0.594667
17	SGDClassifier-StandardScaler-log	No	0.626667
18	BaggingClassifier - SGDClassifier-StandardScaler-log	Yes	0.592889
19	SGDClassifier-StandardScaler-hinge	No	0.624000
20	BaggingClassifier - SGDClassifier-StandardScaler-hinge	Yes	0.597333
21	GaussianNB-StandardScaler	No	0.598222
22	BaggingClassifier - GaussianNB-StandardScaler	Yes	0.610667
23	Grid Search DecisionTreeClassifier Estimator -StandardScaler	Yes	0.637333

Best machine learning model test accuracy of approximately 64% [green colored highlighted box] extracted using the DecisionTreeClassifier algorithm with following ML implement approach/techniques pipeline, StandardScaler, grid search cross validation, grid search best estimator. The worse model test accuracy was from the stochastic gradient descent classifier [red colored highlighted box].

Machine learning Best Model confusion matrix plot for train and test accuracy:



Statistical Significance Test:

Statistical significance plays a pivotal role in statistical hypothesis testing. It is used to determine whether the null hypothesis should be rejected or retained. The null hypothesis is the default assumption that nothing happened or changed. We compared a vanilla Logistic Regression model with the best model from our Grid Search.

The p-value is 0.08225 for a t-score of 1.80012.

There is no significant difference between the two machine learning pipelines (Accept H0)

Conclusion:

We used 31 features for both the home and away teams, plus an indicator for whether the game was played on a neutral court. That equals 63 features available for the model. We experimented extensively with different combinations of numbers of features, (using all 63, using only raw data, using only features we created), different ways we aggregated the data, (actual game stats, averages, home and away averages, differences in stats between the 2 teams) and with different models, (logistic regression, decision tree Classifier, xgboost, etc.). In the end, or best model, which was statistically significant compared to a vanilla logistic regression model, was using the average from the first 80% games of all 63 features for the matchups for both the train and test datasets. It got us to 63.7% accuracy which was better than only picking based on the home team but was a little short of our goal of 64%.

Deep learning – NACC Basketball March Madness Analysis & Prediction

---

REFERENCE:

https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf

https://www.kaggle.com/reisel/how-to-handle-correlated-features

https://www.mdpi.com/1660-4601/15/12/2907/pdf

http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/

https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114

https://elitedatascience.com/feature-engineering

https://towardsdatascience.com/better-heatmaps-and-correlation-matrix-plots-in-python-41445d0f2bec

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

https://towardsdatascience.com/feature-selection-in-python-recursive-feature-elimination-19f1c39b8d15

https://towardsdatascience.com/a-simple-example-of-pipeline-in-machine-learning-with-scikit-learn-e726ffbb6976

https://www.kdnuggets.com/2018/01/managing-machine-learning-workflows-scikit-learn-pipelines-part-3.html

https://www.kdnuggets.com/2018/01/managing-machine-learning-workflows-scikit-learn-pipelines-part-2.html

Project Roles and Responsibility: (Senthil Palani)

1. Data Exploration
2. Feature Engineering
3. Feature Generation
4. Feature Analysis
5. Deep learning –
    a. DEEP LEARNING BASE model - ROOT MEAN SQUARE OPTIMIZER
    b. DEEP LEARNING BASE model - Stochastic gradient descent (SGD) OPTIMIZER
6. Machine Learning: (Optional work)
    a. SVC
    b. LogisticRegression
    c. SGDClassifier

7. Documentation & Review: (Senthil / Saravanan)

Project Roles and Responsibility: (Saravanan Natarajan)

1. Data Exploration
2. Feature Selection
3. Feature Importance
4. Deep learning –
    a. DEEP LEARNING BASE model
    b. DEEP LEARNING BASE model Without standardization
    c. DEEP LEARNING BASE model - GRID SEARCH BEST PARAM IMPL

5. Machine Learning: (Optional work)
    a. GaussianNB
    b. RandomForestClassifier
    c. KNeighborsClassifier
    d. DecisionTreeClassifier

6. Documentation & Review: (Senthil / Saravanan)