**Readme**

**Feature Selection**

* Dropped the unnecessary columns:
  + University Name
  + Rank
  + Mid-Career Pay
  + Degree Length
  + Region
  + State

University Name is the primary key for the table, Rank is state specific, and Mid-Career Pay is essentially another target variable so we elected to drop those columns right away. While we were initially interested in the difference of Early Career Pay between two-year and four-year degrees, we only had six rows of data associated with two-year degrees. Therefore, we did not use the Degree Length column either. After a few rounds of training the decision was made to also remove the Region and State columns as they may have been confusing the model and decreasing its performance.

The features we ended up keeping were the In-State and Out-of-State tuitions, Room and Board, Total Enrollment count, division, school type (public/private), Make World Better %, Stem %, and all of the diversity percentages.

**Data Preprocessing**

* Assessed the data types
* Encoded labels using Pandas.get\_dummies
* Replaced null values with zeros
* Converted continuous target values to categorical values

After feature selection we assessed the remaining data types and encoded the two object fields using Pandas.get\_dummies. We chose to replace null values with zeros instead of removing entire records. Out of the 907 rows, 29 were missing values for the Make World Better Percentage, 50 were missing for Room and Board, and 27 were missing the Enrollment and Diversity percentages. We elected to convert the target column values (Early Career Pay) to either “Low” (less than $45,000) or “Medium/High” for all other amounts using a lambda function. This was done to prepare the data for use with a classification model and to answer the question of whether we are able to accurately predict which features will result in a low early career pay.

**Model Selection and Benefits/Limitations**

Replace picture with text.

For our model we chose the Easy Ensemble Classifier using Adaptive Boosting.

Benefits

* Uses an ensemble of learners that evaluate previous errors and give those errors extra weight when fitting subsequent classifiers
* Utilizes random bootstrap sampling to help with overfitting
* Utilizes random undersampling to help with class imbalance

Limitations

* Outcome can be harder to interpret
* Slower to train and could significantly increase the amount of resources needed

**Model Training and Optimization**

Keep picture but remove first bullet point

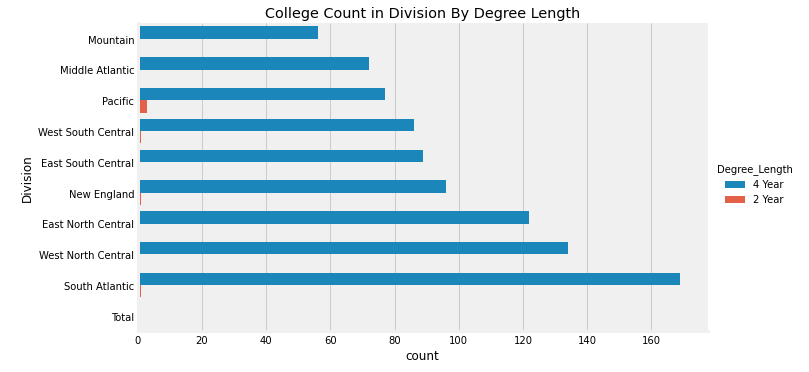
* Adjusted 75/25 train/test split to 80/20
* Increased the number of learners from 10 to 150
* Adjusted the sampling ration from 1.0 to 0.75

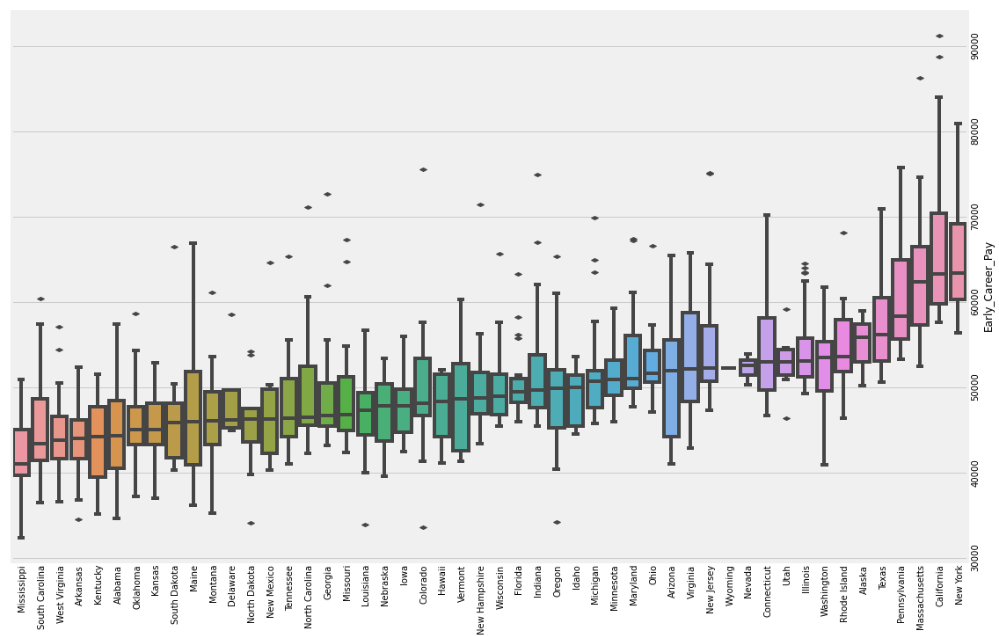
Our model is able to correctly predict 90% of the test targets. With a 97% recall rate, the model correctly identified 37 of the 38 early career salaries that are under $45K. There were 24 false positives (incorrectly predicted as low salary) resulting in a precision rate of 61%.

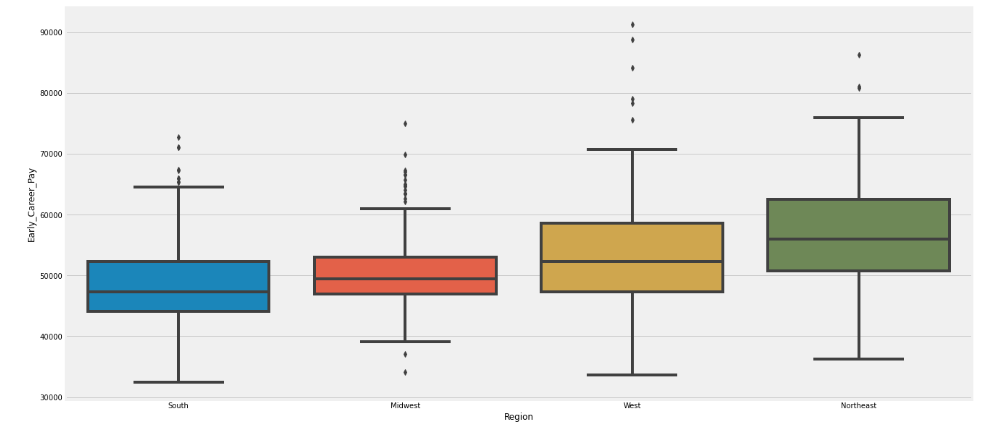
**Slides**

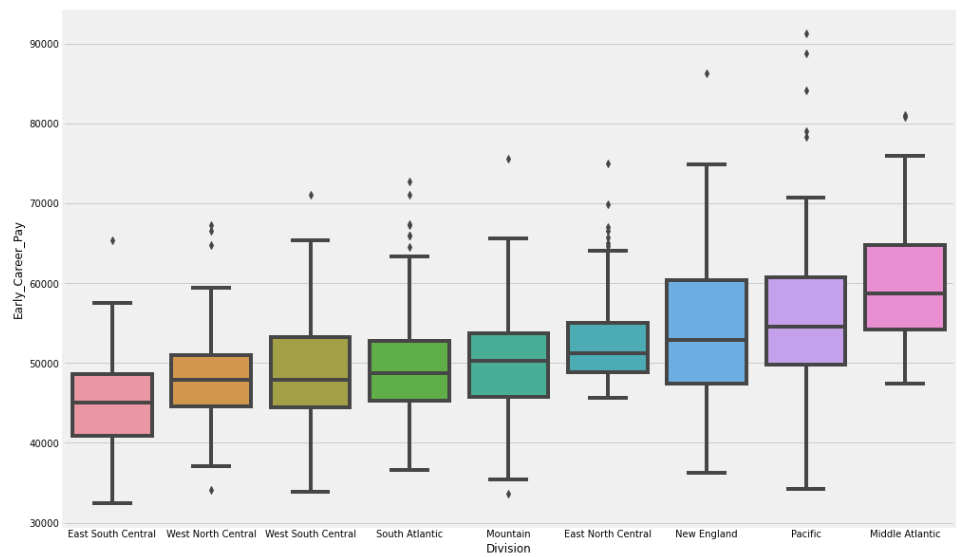
**Feature Selection**

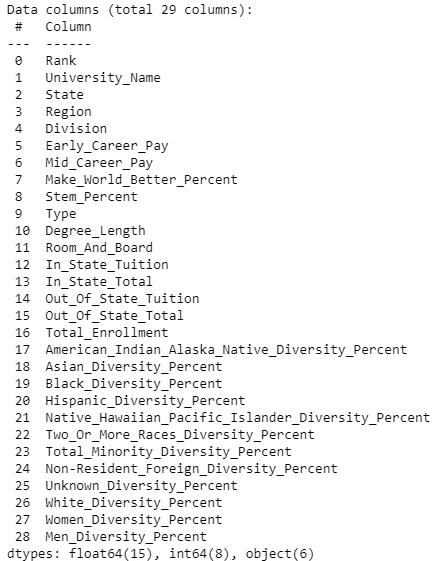
* Visual showing the count issue with the degree length field
* Visuals showing state/region vs division
* Visuals showing the original vs final features. Could maybe fomd something better for this?

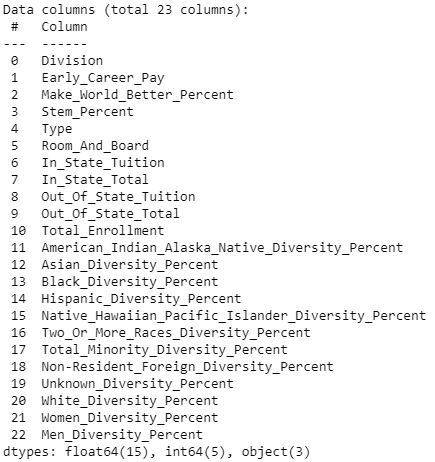












* Dropped columns:
  + University Name
  + Rank
  + Mid-Career Pay
  + Degree Length
  + Region
  + State

**Data Preprocessing**

* Visual showing preprocessing
* Any other visuals we could add to this slide?



* Assessed the data types
* Replaced null values with zeros
* Converted continuous target values to categorical values
* Encoded labels using Pandas.get\_dummies

**Model Selection and Benefits/Limitations**

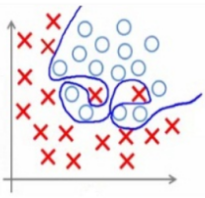
* EasyEnsemble visual/header
* Balancing/scale visual
* Overfitting visual
* Boosting visual



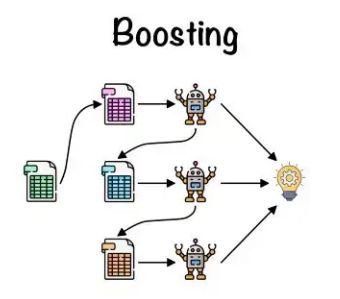
<https://imbalanced-learn.org/stable/references/generated/imblearn.ensemble.EasyEnsembleClassifier.html#ra96f85e96852-1>



<https://medium.com/@erica13chai/coding-problem-balancing-a-scale-11da8d88c823>



<https://moredvikas.wordpress.com/2018/09/24/over-fitting-in-context-of-machine-learning/>



<https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422>

Benefits

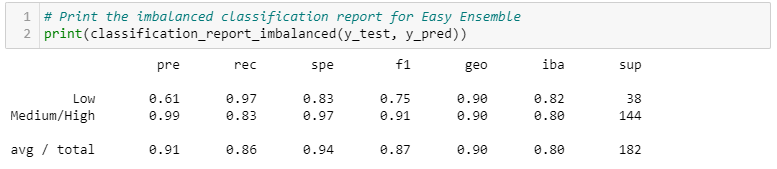
* Uses a bag of boosted learners that evaluate previous errors and give those errors extra weight when fitting subsequent classifiers
* Utilizes random bootstrap sampling to help with overfitting
* Utilizes random undersampling to help with class imbalance

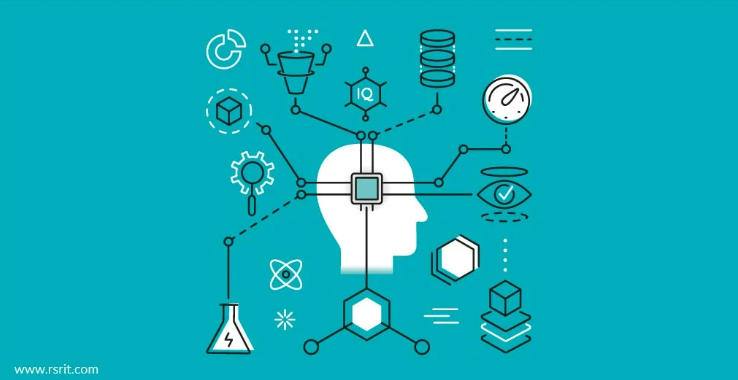
Limitations

* Outcome can be harder to interpret
* Slower to train and could significantly increase the amount of resources needed

**Model Training and Optimization**

* Add confusion matrix if possible, or
* Add an additional slide to show confusion matrix with some fun pictures?



****

<https://blog.rsrit.com/why-a-predictive-model-with-high-accuracy-might-not-be-a-good-model>

**Scripted Notes**

**Feature Selection**

Using our EDA, we were able to get an idea of what features would be important to our model and which wouldn't.  Right off the bat we knew we could remove the Degree Length field as there were only 6 records labeled as 2-year.  (Add EDA picture?)  University Name was our primary key so we dropped that as well.  Mid-Career was essentially another target field so that was also dropped.

After a few rounds of training the decision was made to also remove the Region and State columns as they may have been confusing the model?   pictures of Division/Region/State EDA side by side?

The features we ended up keeping were the In-State and Out-of-State tuitions, Room and Board, Total Enrollment count, division, school type (public/private), Make World Better %, Stem %, and all of the diversity percentages.

**Data Preprocessing**

After deciding which features were important to our model, we assessed the remaining data types and encoded the two object fields (Division and Type) using Pandas get\_dummies.  Nulls were replaced with zeros instead of the rows being removed entirely.  Why?  Which columns?  The last piece of our pre-processing involved using a lambda function to convert our target column amounts into either "Low" or "Medium/High".  Instead of using the Early Career Pay amounts for our model, we converted any amount lower than $45K to "Low" and everything else to "Medium/High" in order to use a classification model to predict schools that result in a low early career salary.

**Model Selection and Benefits/Limitations**

For our model we chose the Easy Ensemble Classifier using Adaptive Boosting. We chose this one because of its ability to learn by evaluating errors in the previous stump and giving higher weights to those errors in the next stumps and because it utilizes random bootstrap sampling to help with overfitting and undersampling to help with class imbalance. One of the limitations of the algorithm is that the outcome can be harder to interpret as it doesn’t include anything similar to the feature importances property used with the Balanced Random Forest Classifier. It may also be slower to train than other algorithms and could significantly increase the amount of resources needed.

**Model Training and Optimization**

We initially used the default 75/25 split for our training and testing sets but later adjusted to 80/20 to increase model performance. This gave us a breakdown of 725 rows of data being used for training and 182 being used for testing. For further optimization, we increased the number of learners in the ensemble from 10 to 150. At some point between 100 and 150 the number of learners failed to provide any further value. We kept the number of estimators within the AdaBoost base estimator at 50. We also adjusted the sampling ratio from 1.0 to 0.75 to decrease the size of the minority class (low salary) within the training set.

Our model is able to correctly predict 90% of the test targets. With a 97% recall rate, the model correctly identified 37 of the 38 early career salaries that are under $45K. There were 24 false positives (incorrectly predicted as low salary) resulting in a precision rate of 61%.

**Outline**

**Feature Engineering**

dropped columns and why

picture of the region EDA - with division and state?

what features were included?

**Pre-process**

Assessed data types and encoded object labels with Pandas get\_dummies

replace nulls with zeros

      why?  didn't remove records

      which columns had nulls?

Convert target column in order to use a classification model - lambda function

**Model choice**

benefits/limitations

      slower to train

      harder to see how the model works

      uses bootstrapping to overcome overfitting problem

how was model trained/optimized?

      trained 725, tested 182

      number of learners in the ensemble

      sampling ratio = 0.75

confusion matrix and accuracy score

Not used:



