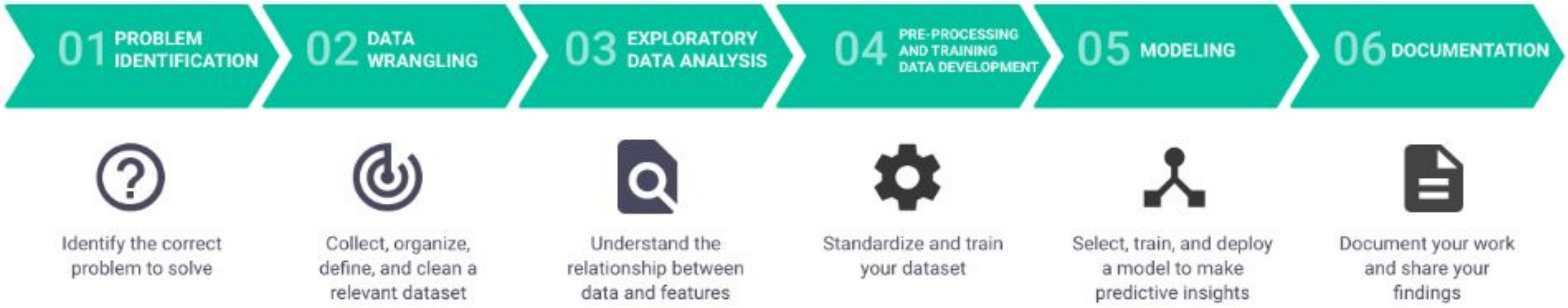
Modelling the Number of Awards Issued for Every 100 Full-time Undergraduate Students



01 PROBLEM IDENTIFICATION

Identify the correct problem to solve

02 DATA WRANGLING

Collect, organize, define, and clean a relevant dataset

03 EXPLORATORY DATA ANALYSIS

Understand the relationship between data and features

04 PRE-PROCESSING AND TRAINING DATA DEVELOPMENT

Standardize and train your dataset

05 MODELING

Select, train, and deploy a model to make predictive insights

06 DOCUMENTATION

Document your work and share your findings

Problem Statement

Question:

Can we develop a model that classifies higher education institutions based on the number of awards issued per 100 full-time undergraduate students with at least 90% accuracy?

Rationale:

In the United States, selecting a higher education institution involves considering factors like degree offerings, institutional control (private vs. public), and financial aid availability. I propose a classification model to predict awards issued per 100 students with high accuracy.

Data Wrangling

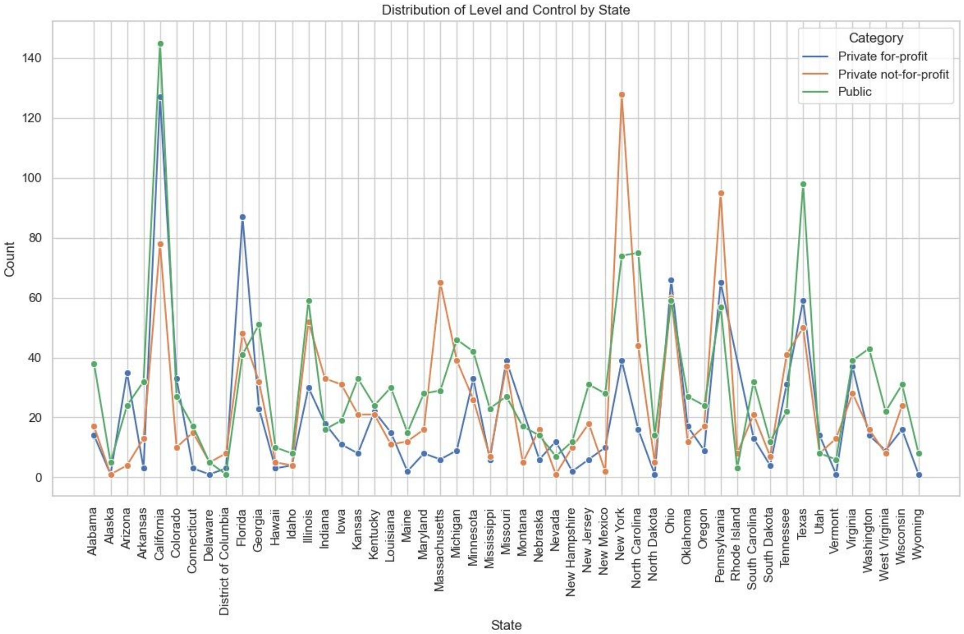
Data Source: Obtained from Kaggle (link: dataset)

Data Cleaning:

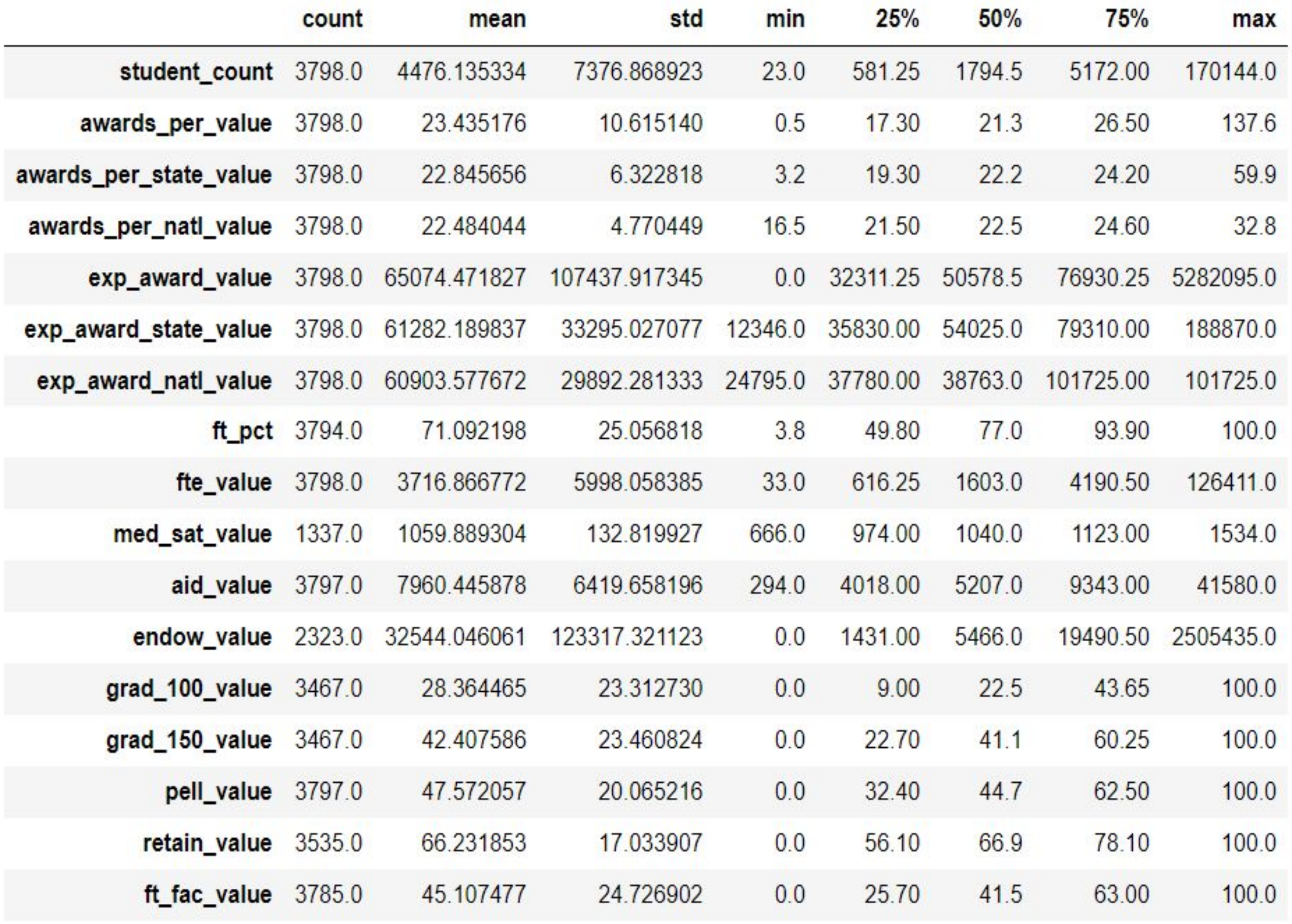
- **Duplicates:** Removed duplicate entries.
- **Missing:** Filled missing values in ‘flagship’ and ‘hbcu’ columns.
- **Imputation:** Imputed mean for numeric columns with missing values.
- **Outliers**: Removed using Interquartile Range (IQR) method.
- **Encoding**: OneHotEncoding for non-numeric variables (‘level’, ‘control’, ‘basic’, ‘hbcu’, ‘flagship’).

Exploratory Data Analysis

- **Geographical Distribution**:
  - California had the highest number of institutions (350).
  - Lowest numbers in District of Columbia, Rhode Island, Delaware, Wyoming, and Alaska.
- **Institution Control**:
  - More 4-year institutions than 2-year institutions.

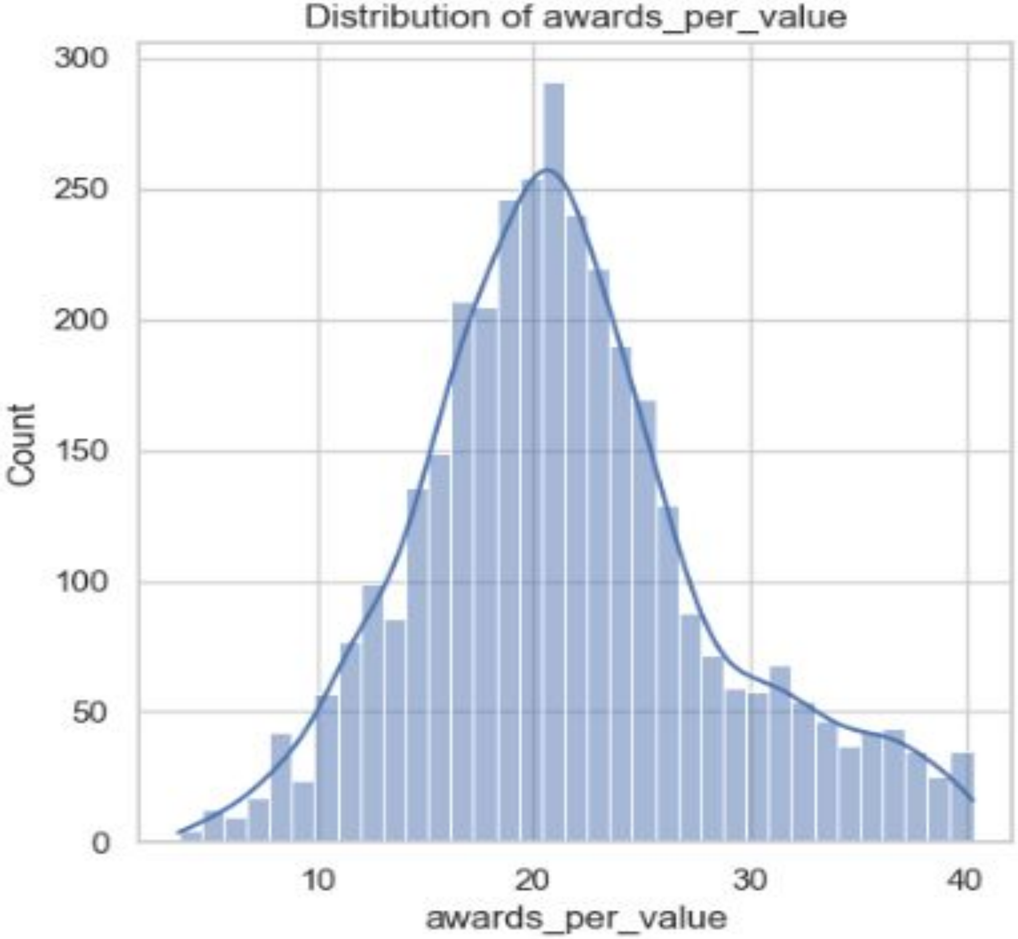


Some Descriptive Statistics

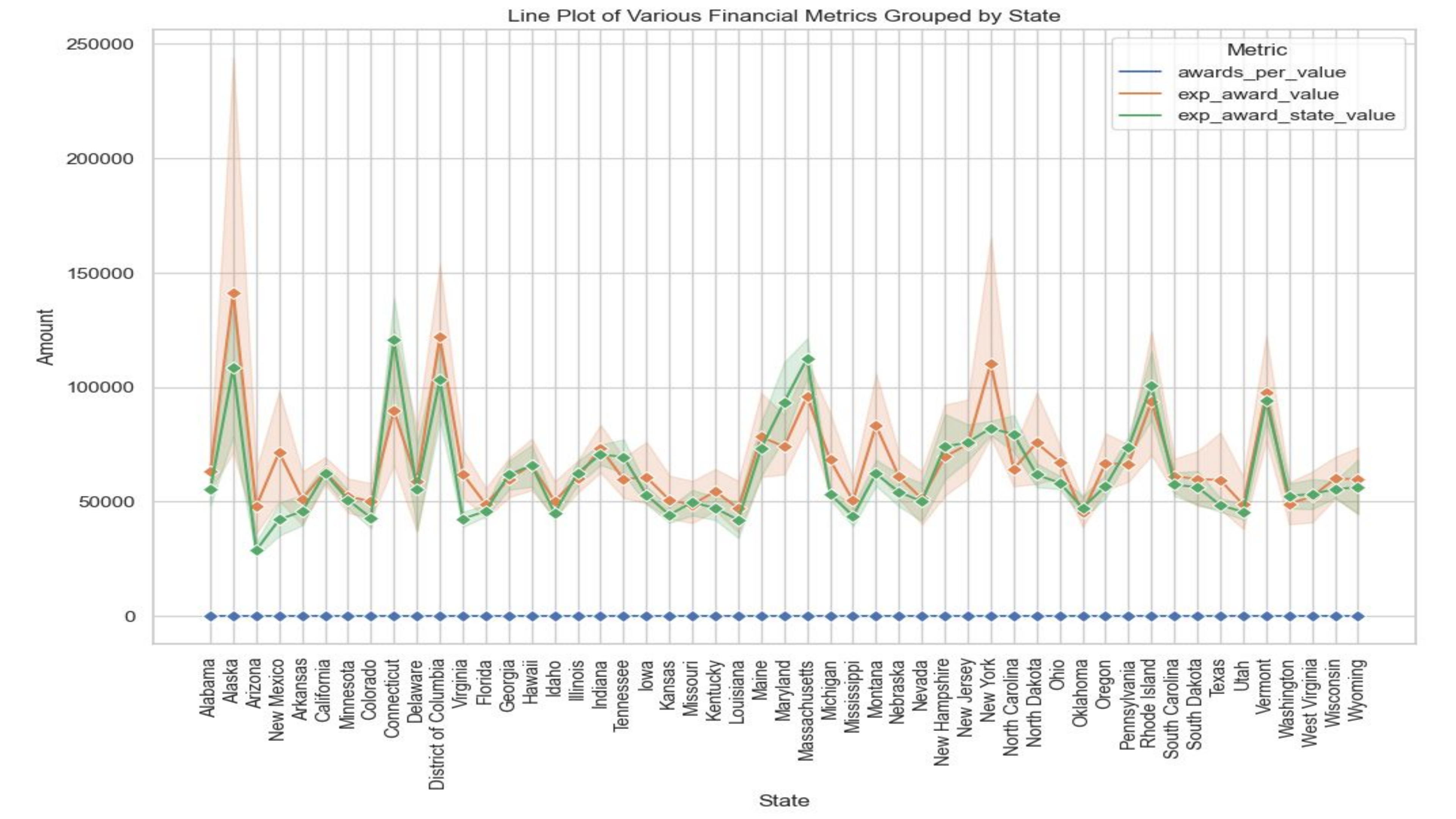


	count	mean	std	min	25%	50%	75%	max
student_count	3798.0	4476.135334	7376.868923	23.0	581.25	1794.5	5172.00	170144.0
awards_per_value	3798.0	23.435176	10.615140	0.5	17.30	21.3	26.50	137.6
awards_per_state_value	3798.0	22.845656	6.322818	3.2	19.30	22.2	24.20	59.9
awards_per_natl_value	3798.0	22.484044	4.770449	16.5	21.50	22.5	24.60	32.8
exp_award_value	3798.0	65074.471827	107437.917345	0.0	32311.25	50578.5	76930.25	5282095.0
exp_award_state_value	3798.0	61282.189837	33295.027077	12346.0	35830.00	54025.0	79310.00	188870.0
exp_award_natl_value	3798.0	60903.577672	29892.281333	24795.0	37780.00	38763.0	101725.00	101725.0
ft_pct	3794.0	71.092198	25.056818	3.8	49.80	77.0	93.90	100.0
fte_value	3798.0	3716.866772	5998.058385	33.0	616.25	1603.0	4190.50	126411.0
med_sat_value	1337.0	1059.889304	132.819927	666.0	974.00	1040.0	1123.00	1534.0
aid_value	3797.0	7960.445878	6419.658196	294.0	4018.00	5207.0	9343.00	41580.0
endow_value	2323.0	32544.046061	123317.321123	0.0	1431.00	5466.0	19490.50	2505435.0
grad_100_value	3467.0	28.364465	23.312730	0.0	9.00	22.5	43.65	100.0
grad_150_value	3467.0	42.407586	23.460824	0.0	22.70	41.1	60.25	100.0
pell_value	3797.0	47.572057	20.065216	0.0	32.40	44.7	62.50	100.0
retain_value	3535.0	66.231853	17.033907	0.0	56.10	66.9	78.10	100.0
ft_fac_value	3785.0	45.107477	24.726902	0.0	25.70	41.5	63.00	100.0

Looking at the target variable ‘awards_per_value’, the mean is low while the standard deviation is high. This is because the description is based on the original data with minimal cleaning, e.g. the outliers have not been removed.



The distribution of awards when outliers were removed appears to be normal.


Line Plot of Various Financial Metrics Grouped by State
Metric
awards_per_value
exp_award_value
exp_award_state_value
Amount
250000
200000
150000
100000
50000
0
State
Alabama
Alaska
Arizona
New Mexico
Arkansas
California
Minnesota
Colorado
Connecticut
Delaware
District of Columbia
Virginia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Tennessee
Iowa
Kansas
Missouri
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Mississippi
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Texas
Utah
Vermont
Washington
West Virginia
Wisconsin
Wyoming

Pre-Processing and Modelling

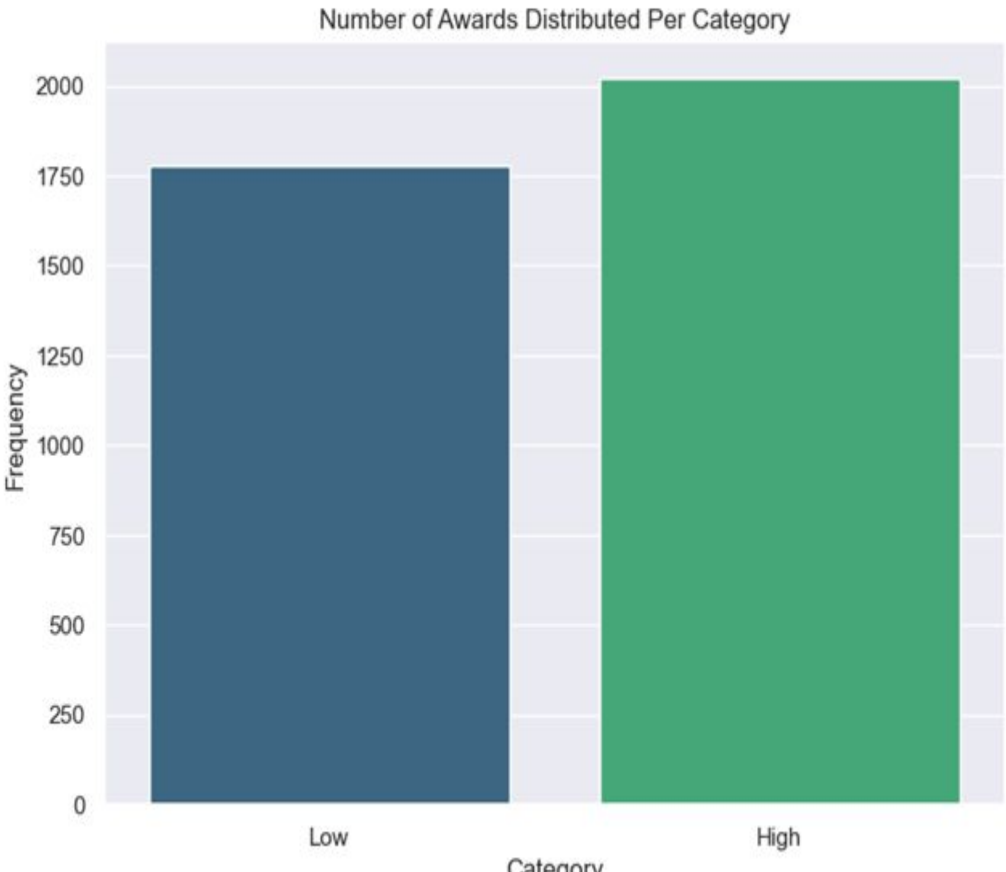
Data Transformation:

- Created ‘num_awards_given’ column by binning ‘awards_per_value’.
- The classes seem to be somewhat imbalanced. Hence, model performance will be assessed through F1-Score reporting.

Data Splitting and Scaling:

- Split data into training (80%) and testing (20%) sets.
- Scaled using StandardScaler.

Data Balanced?

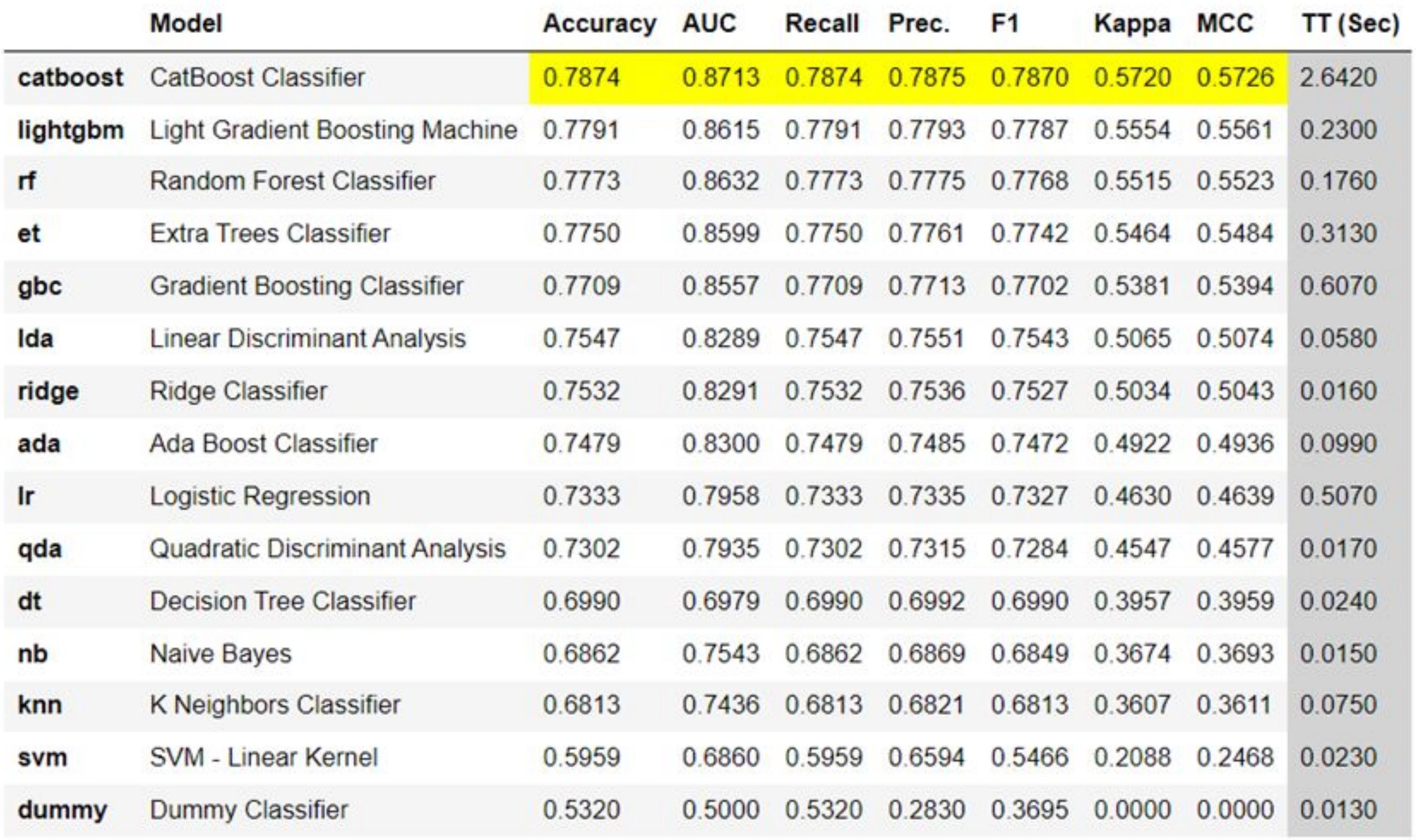


Training/Testing using PyCaret

```
import pycaret
pycaret.__version__
from pycaret.classification import *

s = setup(df, target = 'num_awards_given', session_id = 123)

# Train and select a model
best = compare_models()

#Evaluate a trained model
evaluate_model(best)

# Predict on test data
pred_test = predict_model(best)

# Predict on new data
new_data = df.copy().drop('num_awards_given', axis=1)
predictions = predict_model(best, data = new_data)
```

PyCaret provides an easy way to compare multiple machine learning models across various metrics and selects the best model with low amount of coding. The entire machine learning pipeline is automated making it easier to focus on interpreting results.Link: PyCaret reference

Model Performance



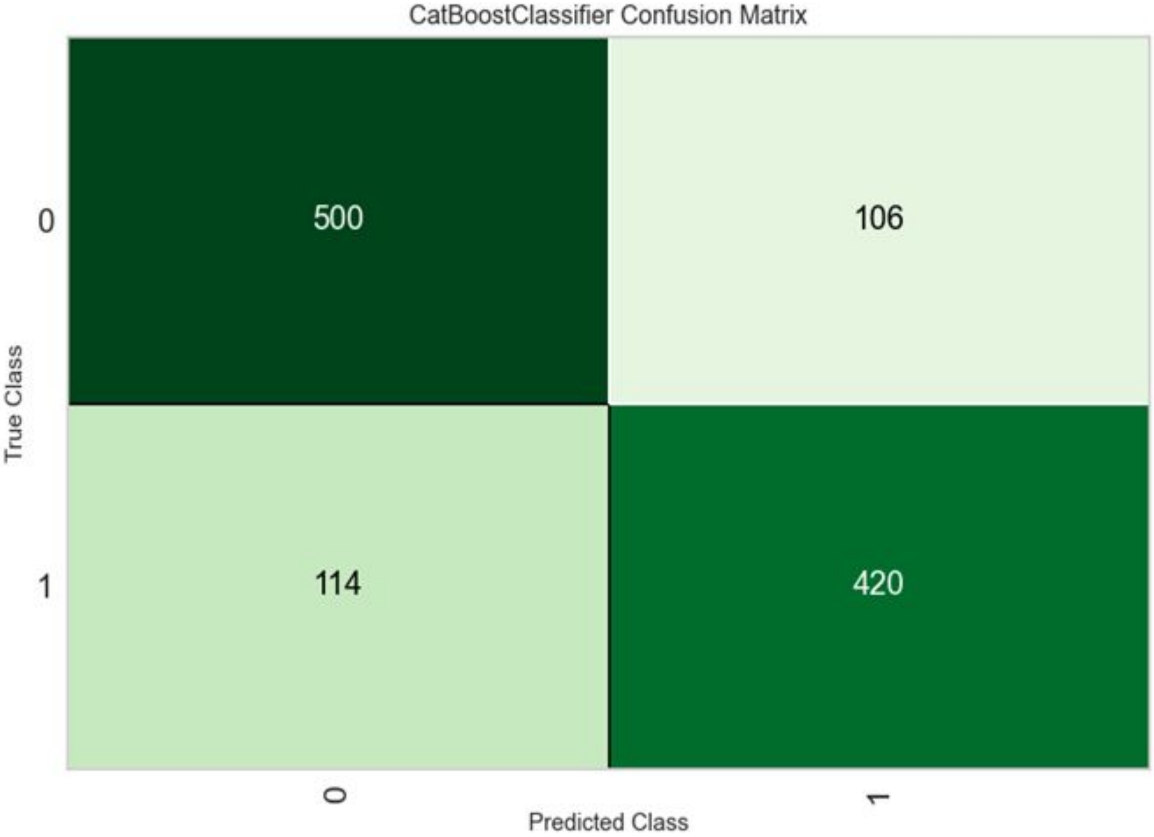
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.7874	0.8713	0.7874	0.7875	0.7870	0.5720	0.5726	2.6420
lightgbm	Light Gradient Boosting Machine	0.7791	0.8615	0.7791	0.7793	0.7787	0.5554	0.5561	0.2300
rf	Random Forest Classifier	0.7773	0.8632	0.7773	0.7775	0.7768	0.5515	0.5523	0.1760
et	Extra Trees Classifier	0.7750	0.8599	0.7750	0.7761	0.7742	0.5464	0.5484	0.3130
gbc	Gradient Boosting Classifier	0.7709	0.8557	0.7709	0.7713	0.7702	0.5381	0.5394	0.6070
lda	Linear Discriminant Analysis	0.7547	0.8289	0.7547	0.7551	0.7543	0.5065	0.5074	0.0580
ridge	Ridge Classifier	0.7532	0.8291	0.7532	0.7536	0.7527	0.5034	0.5043	0.0160
ada	Ada Boost Classifier	0.7479	0.8300	0.7479	0.7485	0.7472	0.4922	0.4936	0.0990
lr	Logistic Regression	0.7333	0.7958	0.7333	0.7335	0.7327	0.4630	0.4639	0.5070
qda	Quadratic Discriminant Analysis	0.7302	0.7935	0.7302	0.7315	0.7284	0.4547	0.4577	0.0170
dt	Decision Tree Classifier	0.6990	0.6979	0.6990	0.6992	0.6990	0.3957	0.3959	0.0240
nb	Naive Bayes	0.6862	0.7543	0.6862	0.6869	0.6849	0.3674	0.3693	0.0150
knn	K Neighbors Classifier	0.6813	0.7436	0.6813	0.6821	0.6813	0.3607	0.3611	0.0750
svm	SVM - Linear Kernel	0.5959	0.6860	0.5959	0.6594	0.5466	0.2088	0.2468	0.0230
dummy	Dummy Classifier	0.5320	0.5000	0.5320	0.2830	0.3695	0.0000	0.0000	0.0130

CatBoost model is the highest, best performing model with F1-score = 78.7%.

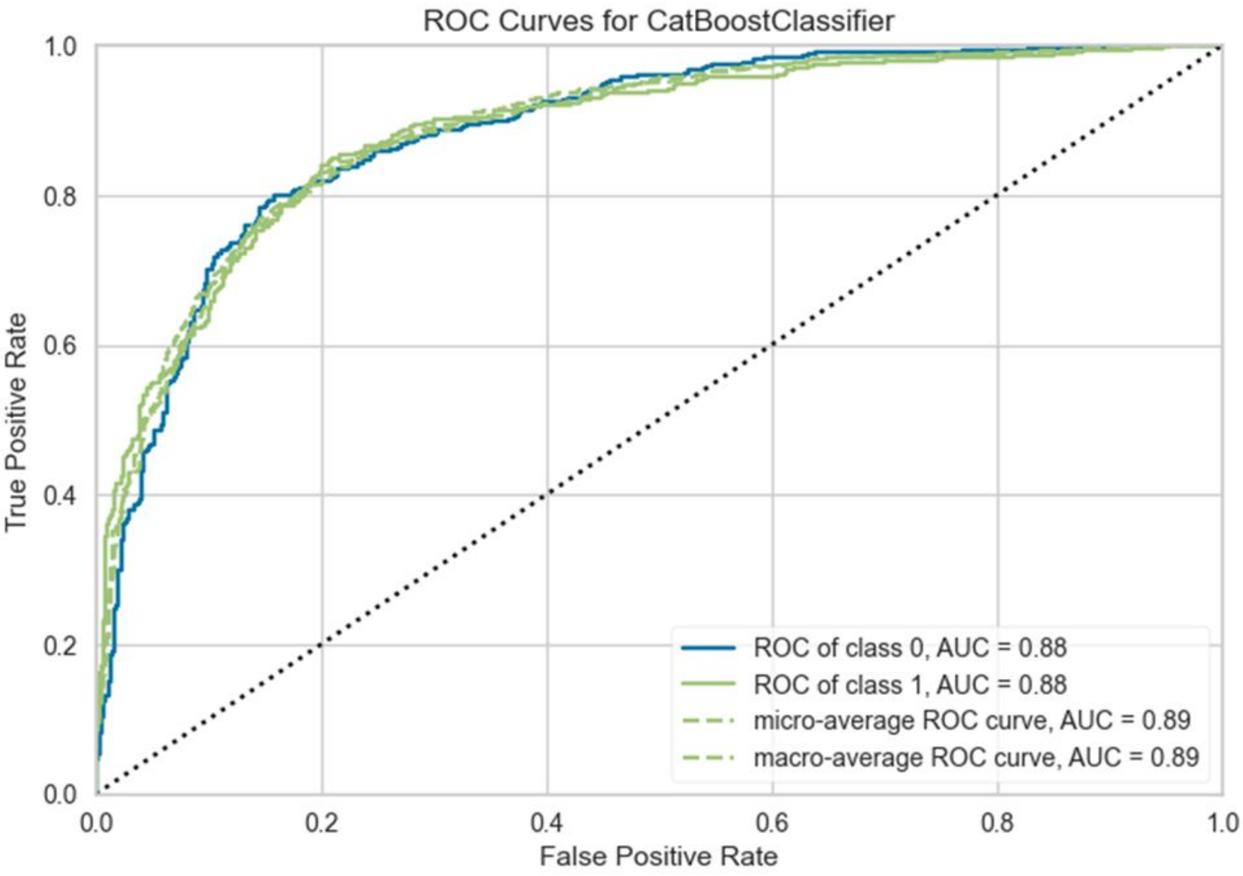
CatBoost Model is also the most expensive model based on the table.

CatBoostClassifier Model Analysis

Based on the confusion matrix plot, the model has a high true positive rate while it maintains a low false positive rate.

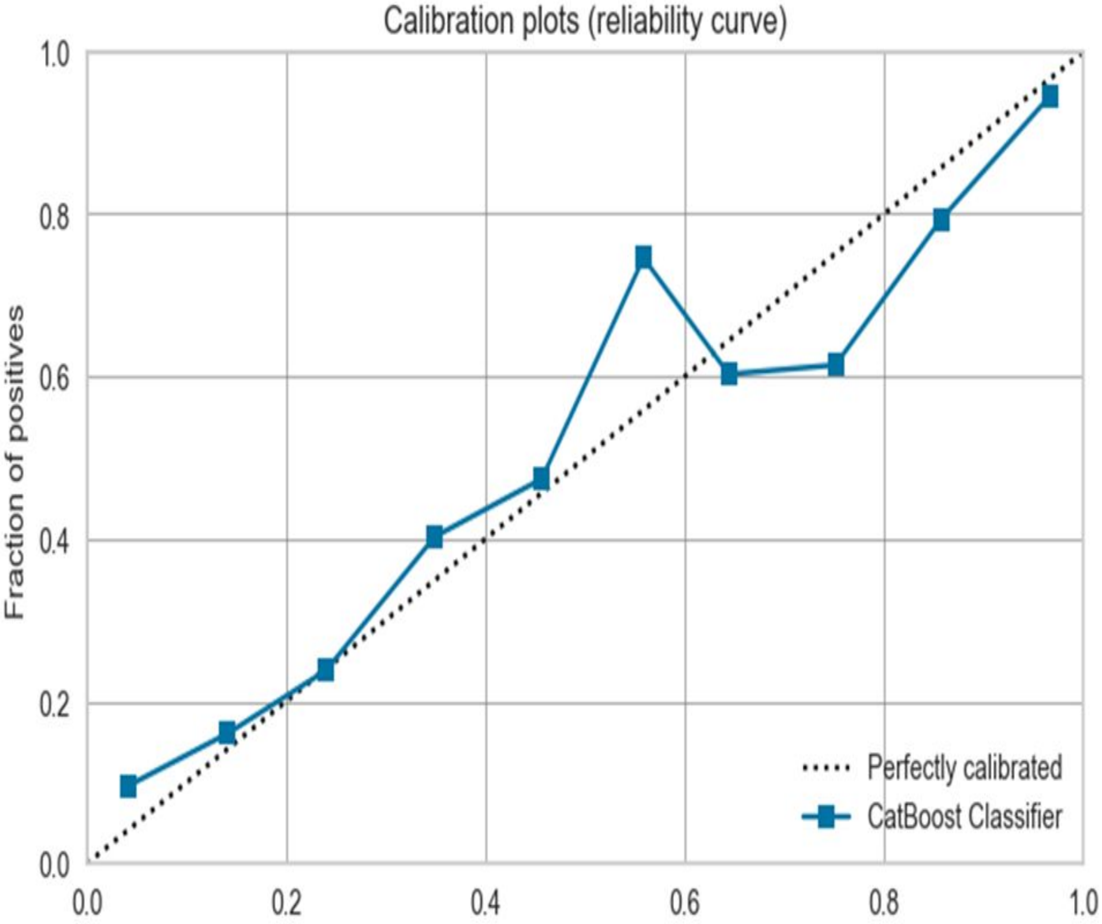


ROC Curve



- The CatBoostClassifier model performs significantly better than random guessing.
- The micro-average ROC curve reflects the model’s overall performance across all classes and samples. With AUC = 0.89, the model is slightly better performing when considering the overall dataset. While the macro-average AUC suggests a balanced performance across both classes without being skewed by class imbalance.

CatBoost Model’s Reliability Curve



- The model is well calibrated, especially in the extreme ends (low and high probabilities). There are some discrepancies in the mid-range probabilities where the model under/overestimates the likelihood of positive outcomes.
- Though the model’s probability predictions are mostly reliable, there are areas where calibration could be improved.

Hyperparameter tuning

Randomized search cross validation was use to tune the model. Here’s the code:

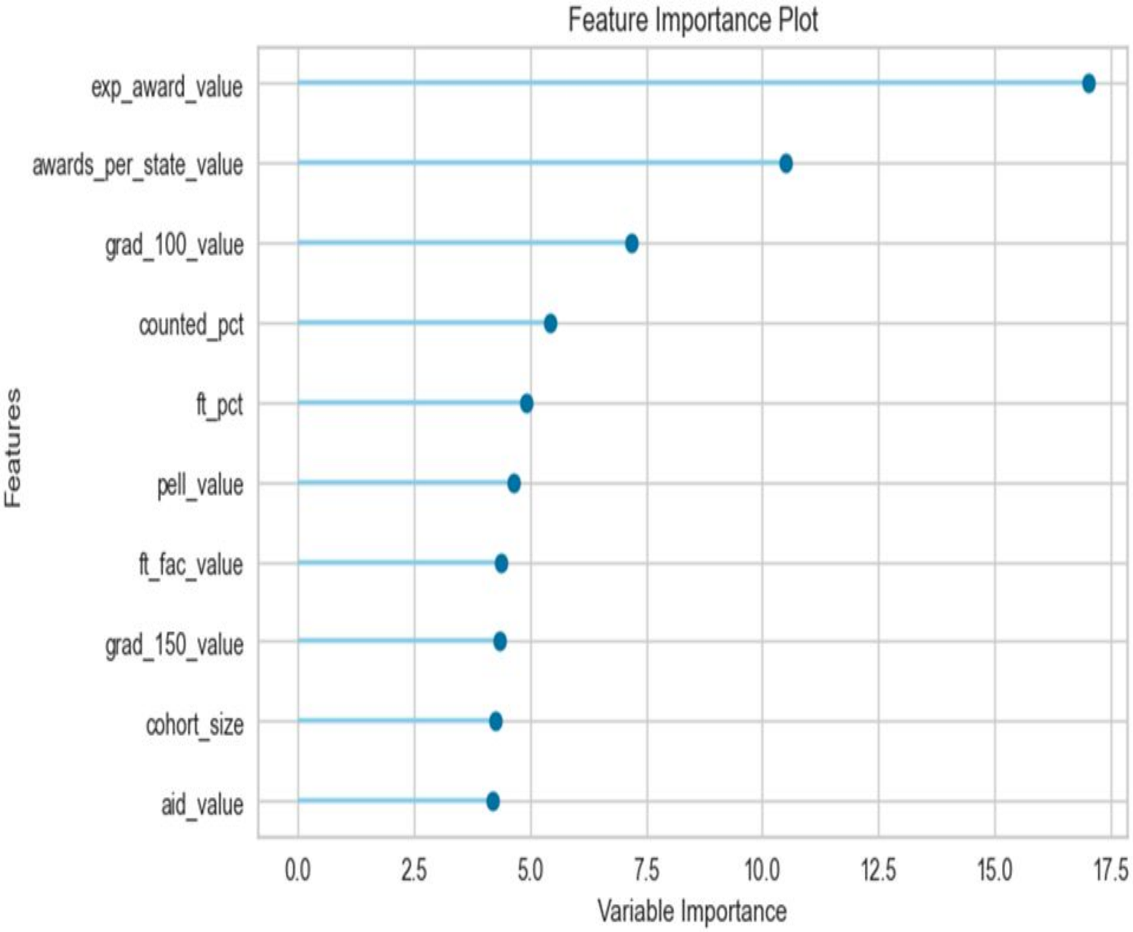
```
cb_search = RandomizedSearchCV(estimator=cb_model, param_distributions=cb_param_dist,
                               scoring='accuracy', n_iter=50, cv=5, random_state=42)
cb_search.fit(X_train_scaled, y_train)
```

Best Parameters: {'depth': 9, 'iterations': 238, 'l2_leaf_reg': 1, 'learning_rate': 0.0401435087930859}

```
              precision    recall  f1-score   support

        High       0.79      0.86      0.83       398
         Low       0.83      0.75      0.79       362

    accuracy                           0.81       760
   macro avg       0.81      0.81      0.81       760
weighted avg       0.81      0.81      0.81       760
```

- With F1-Score = 0.83, the model has a good balance between and recall for all instances classified as ‘High’. This indicates effective performance of the model at classifying True High instances.
- There is a slightly lower F1-score for predicting True Low instances.
- Generally, the model is reliable and robust at correctly classifying instances in both classes.

Feature Importances



The feature importance plot clearly indicates that financial metrics (such as exp_award_value, awards_per_state_value, and aid_value) and student success indicators (like grad_100_value and grad_150_value) are paramount in predicting the number of awards issued per 100 full-time undergraduate students. The contributions from faculty and the structure of student enrollment (ft_fac_value, ft_pct and cohort_size) also play significant roles, providing a comprehensive view of the factors influencing award distribution in higher education institutions.

Conclusion

I begin the project by asking whether I can develop a model that classifies higher education institutions based on the number of awards issued per 100 full-time undergraduate students with at least 90% accuracy. Based of the best performing model,

- We can classify institutions with high number of awards with an accuracy of 83%.
- We can classify institutions with that have a low number of awards issued with 79 % accuracy.

Although the best model falls short of my target of at least 90% accuracy, the current CatBoost model is effective.

Recommendations

1. **Model Tuning:** Explore additional hyperparameters and ensemble methods to improve model accuracy.

2. **Classes:** Run a simple clustering analysis to try and identify potential classes. Just to get an overview (as opposed to the current manual process by which the classes were obtained).

3. **Deployment:** Developing an application for prospective students and families to use the predictive model. Will be using Streamlit.