# Final Project

Joseph Nelson Farrell & Michael Massone
DS 5220 Supervised Machine Learning
Northeastern University
Professor Steven Morin, PhD

# Introduction

# Objective

The company has segmented its existing customers into four distinct classes based on the marketing strategy that works best for them. These strategies include text messaging (A), physical mailers (B), weekly emails (C), and quarterly emails(D).

The marketing team has determined that these strategies have proven to be effective for their corresponding customer segments; however, they tend to have the opposite effect on customers outside those groups, potentially driving them away. For example, customers who have preferred quarterly emails, have been more likely to label promotional emails as spam when the frequency of emails is increased, while those who have preferred weekly emails are less likely to convert when email frequency decreases.

The company aims to generalize this segmentation to new customers by using available information from prospective customers who have signed up for promotional materials. Their goal is to develop a predictive model that can accurately assign new customers to the appropriate marketing strategy class, ensuring that each customer receives the communication style most likely to resonate with them, while avoiding strategies that might alienate them. This classification model must carefully consider customer features to predict which marketing strategy will yield the best results, helping the company optimize its outreach and improve overall customer engagement and retention.

## Class: A

**Text Messages**

## Class: B

**Physical Mailers**

## Class: C

**Weekly Emails**

## Class: D

**Quarterly Emails**

# Reducing From Multiclass to Binary Classification

An initial exploration of model performance was conducted across all combinations of binary classification schemes for the four target classes provided by the stakeholders. After discussing the complexities and challenges associated with multi-class classification, the stakeholders decided to prioritize binary classification over a multi-class approach.

The decision was driven by the marketing team's analysis, which revealed that class A could be treated as a subclass of B, and class C as a subclass of D. This reframing reduced the problem to a binary classification task, offering a simpler solution with significantly lower time and resource demands, albeit at the potential expense of a more nuanced classification schema.

**Class: B**

Physical Mailers

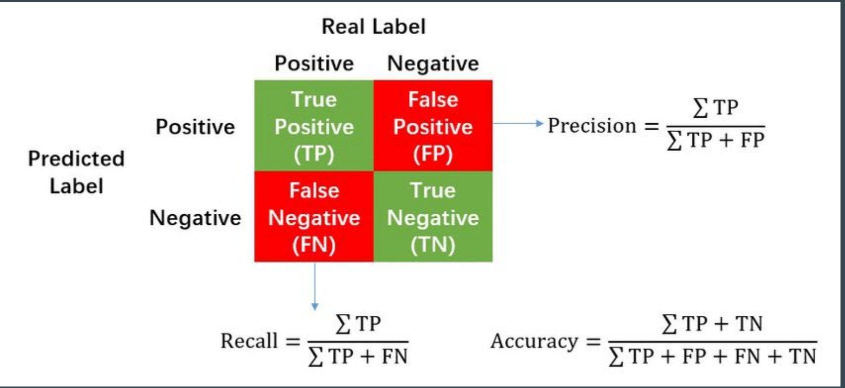
**Class: D**

Quarterly Emails

# Defining and Achieving Success

Physical mailers are significantly more costly than text messages and emails (which are virtually free). Stakeholders have determined that misclassifying Class D as Class B would be financially wasteful. Additionally, physical mailers are unlikely to influence the spending habits of Class D individuals. Conversely, individuals in Class B are more likely to be influenced by email correspondence due to the ubiquity of email in modern communication.

Given these considerations, the primary objective of the model is to minimize false positives for Class B. This means the model will prioritize precision with respect to Class B (treated as the positive class during training).




Cred:https://www.researchgate.net/figure/Calculation-of-Precision-Recall-and-Accuracy-in-the-confusion-matrix_fig3_336402347

Precision and recall are defined as follows:

- Precision = TP / TP + FP
- Recall = TP / TP + FN

Since precision is sensitive to false positives and recall is sensitive to false negatives, we will prioritize precision to align with the stakeholders' goals.

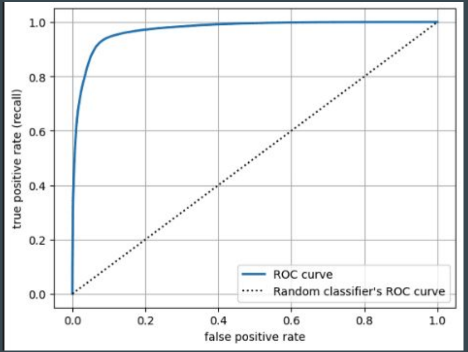
During model selection, we will focus on the ranking metric average precision. For tuning the classification threshold, we will emphasize the classification metric precision

Overall success will be achieved with a precision of class B greater than 0.80.

# Ranking Metrics

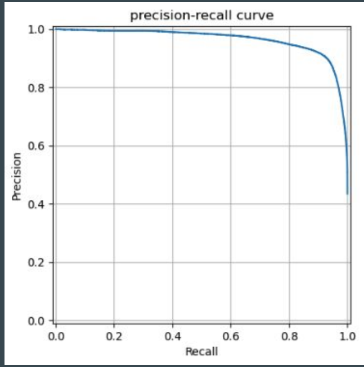
Throughout this presentation we will be using Average Precision and ROC AUC ranking metrics.

## ROC AUC




- Measures the tradeoff between **True Positive Rate** and **False Positive Rate** over the range of classification thresholds.
- Perfect classifier has an ROC AUC = 1.00
- Random classifier has ROC AUC = 0.50

## Average Precision




- Measures the tradeoff between **Precision** and Recall over the range of classification thresholds.
- Perfect classifier has an Avg. Precision = 1.00
- Random classifier has Avg. Precision = 0.50

1. Dataset Review
2. Feature Selection
3. Model Exploration

# Dataset Review

# Data

**Dataset** : Customer Segmentation

**Origins** : https://www.kaggle.com/datasets/vetrirah/customer?select=Train.csv

**Description** : The dataset contains information about customers of an automobile company segmented into 4 classes (target)

## Size Full (with all classes):

- Size: (8068, 11)
- Instances: 8068
- Attributes: 11

## Size Filtered (contains only classes B & D):

- Size: (3300, 11)
- Instances: 3300
- Attributes: 11

# The Target Variable

The column name of the target variable is Segmentation.

- There are **0** missing values in the target variable column.
- As a result, **0** observations (rows) are dropped.
- Dataset size: (3300, 11)
- The datatype of the target is a string with 4 values:
  - Value: B ~ Proportion: 0.450303
  - Value: D ~ Proportion: 0.549697
- The categories appear relatively balanced.
- These values correspond to customer segments used in outreach.

# Attributes

| Attribute Name | AttributeType | Percent Missing Values | ML Attribute Designation |
|---|---|---|---|
| index | Numerical - Discrete | 0.0% | non_ML |
| ID | Numerical - Discrete | 0.0% | non_ML |
| Gender | Categorical - Nominal | 0.0% | ML |
| Ever_Married | Categorical - Nominal | 1.9% | ML |
| Age | Numerical - Discrete | 0.0% | ML |
| Graduated | Categorical - Nominal | 1.1% | ML |
| Profession | Categorical - Nominal | 2.1% | ML |
| Work_Experience | Numerical - Discrete | 12.1% | ML |
| Spending_Score | Categorical - Ordinal | 0.0% | ML |
| Family_Size | Numerical - Discrete | 4.6% | ML |
| Var_1 | Categorical - Nominal | 0.9% | ML |

# ML Attribute Selection

## ML Attributes

Numerical Attributes:
- Age
- Work_Experience
- Family_Size

Categorical Attributes:
- Gender
- Ever_Married
- Graduated
- Profession
- Spending_Score
- Var_1

Total Attributes:
- 9

## Non-ML Attributes

Non-ML Attributes List:
- ID
- Index

Missingness Drop List:
- None
- No attributes missing > 20 % of the observations

ML Attributes Drop List:
- None
- No attributes were identified during EDA to exclude.

Total Attributes:
- 2

# Attribute Transformation

## Categorical

Data Imputation:

- Fill missing values with the most frequent

Data Encoding:

- Target encoding
- Each category is encoded based on an estimate of the target average for that category. The encoding mixes the overall target mean with target mean given the value of the category.

Data Scaling:

- Standard scaling
- De-mean
- Whiten

## Numerical

Data Imputation:

- Fill missing values with the mean.

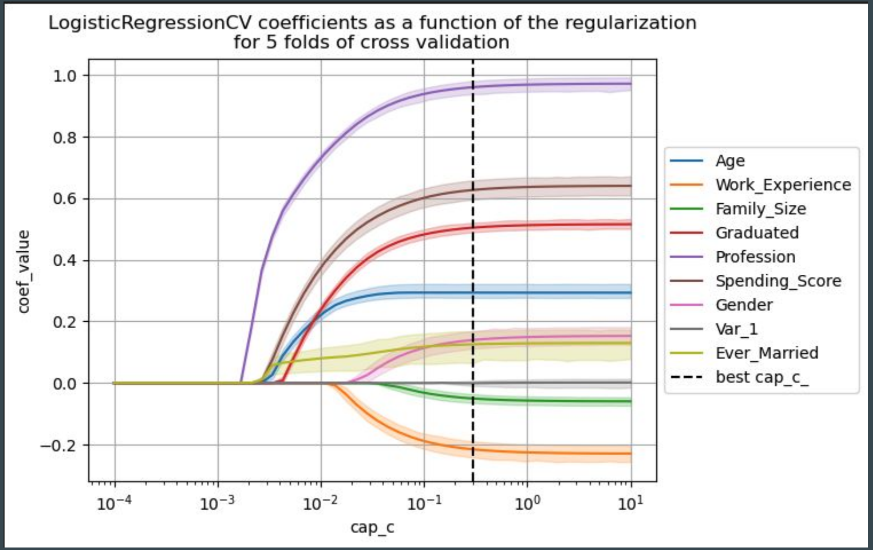
Data Scaling:

- Standard scaling
- De-mean
- Whiten

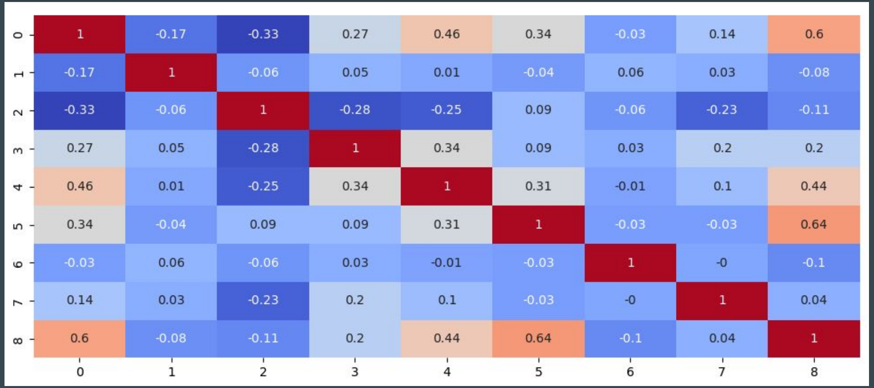
# Feature Selection

# Reducing Model Complexity with LASSO & VIF

- The VIF scores demonstrate multicollinearity within the data.
- The correlation matrix confirms correlated attributes.
- This results in the instability seen in the LASSO coefficient paths.
- We can remove the attribute the highest VIF score and re-instantiate the model.
- The attribute removed on iteration 1:
  - **Ever_Married.**
- We will iteratively repeat this process until all the attributes have a VIF score that is close 1.

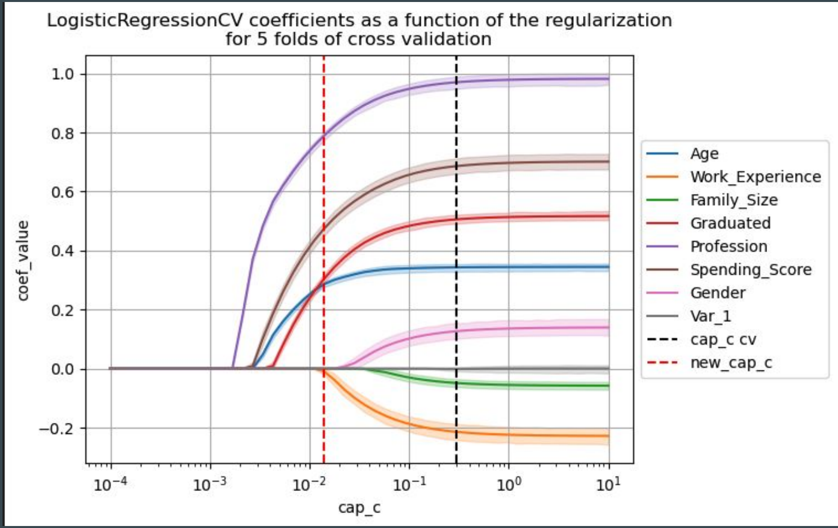



| ATTRIBUTES | VIF |
|---|---|
| Gender | 1.03 |
| Work_Experience | 1.06 |
| Var_1 | 1.09 |
| Graduated | 1.22 |
| Family_Size | 1.30 |
| Profession | 1.47 |
| Spending_Score | 1.81 |
| Age | 1.90 |
| Ever_Married | 2.44 |

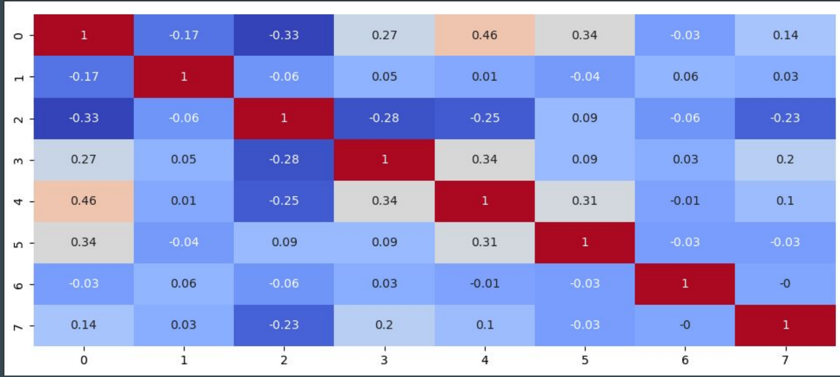


# Reducing Model Complexity with LASSO & VIF

- Removing attributes with high VIF scores until all attributes had VIF of close to 1 improved the stability of the LASSO paths.
- The correlation matrix confirms that the correlation between the attributes has improved.
- Number of attributes removed via VIF process:
  - 1
- Attributes remaining:
  - 8
- We can further simplify our model by trading off 1 standard deviation in log loss and move towards more regularization.
- This results in 4 non-zero attributes
  - Age, Graduated, Profession, Speding_Score




| ATTRIBUTES | VIF | Removed via VIF | Removed via Tradeoff |
|---|---|---|---|
| Gender | 1.01 | Ever_Married | Work_Experience |
| Work_Experience | 1.06 | | Family_Size |
| Var_1 | 1.09 | | Gender |
| Graduated | 1.22 | | Var_1 |
| Spending_Score | 1.26 | | |
| Family_Size | 1.30 | | |
| Profession | 1.43 | | |
| Age | 1.56 | | |



# Attribute Selections

| Attribute Name | AttributeType |
|---|---|
| Age | Numerical - Discrete |
| Graduated | Categorical - Nominal |
| Profession | Categorical - Nominal |
| Spending_Score | Categorical - Ordinal |

## Rational

**Reduced Risk of Overfitting**: Fewer features reduce the complexity of the model, making it less prone to overfitting. With fewer features, the model will likely generalizes better to unseen data since it captures only the most relevant patterns, rather than noise or spurious correlations.

**Model Interpretability**: A simpler model with only 4 features is easier to interpret and explain to stakeholders - even in situation when feature importance is not part of the analysis.

**Lower Variance in Predictions**: More complex models are often more sensitive to small changes in the training data, which can lead to higher variance and inconsistent predictions.

**Computational Cost and Inference Time**: A simpler model takes less time to train and evaluate, making it more efficient for real-world applications. Inference time is also faster, which can be important in production systems or when deploying a model.

**Maintenance:** Simpler models are easier to maintain and update, especially if the feature set changes. With fewer features, the data collection pipeline is simpler, reducing potential errors and data management overhead.

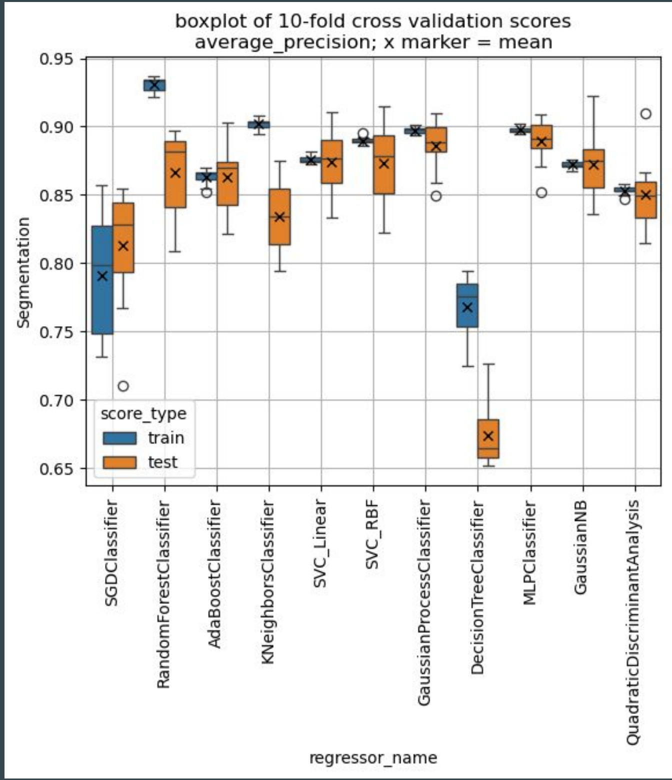
# Model Exploration

# Training Composite Estimators

- 11 composite estimators we trained with default instantiations.
- The most performant were shortlisted for further hyperparameter tuning.
- Given a balanced dataset and the concerns of the stakeholders about false positives, we will be making our determination based on average precision.

**All Estimators:**
SGDClassifier
RandomForestClassifier
AdaBoostClassifier
KNeighborsClassifier
SVC_Linear
SVC_RBF
GaussianProcessClassifier
DecisionTreeClassifier
MLPClassifier
GaussianNB
QuadraticDiscriminantAnalysis

**Shortlisted Estimators:**
RandomForestClassifier
SVC_Linear
SVC_RBF
MLPClassifier



# Estimator Selections

| Estimator Name: | Estimator Model Family: | Avg_precision |
|---|---|---|
| RandomForestClassifier | Ensemble (Bagging) | **Train**: 0.930920<br>**Test**: 0.866272 |
| SVC_Linear | Kernel-based method | **Train**: 0.875855<br>**Test**: 0.874027 |
| SVC_RBF | Kernel-based method | **Train**: 0.889699<br>**Test**: 0.873479 |
| MLPClassifier | Neural Network | **Train**: 0.897438<br>**Test**: 0.889064 |

## Rational

Our initial selection of estimators demonstrated a wide range of results on the key metrics of avg_precision and ROC-AUC. The Decision Tree classifier was the worst-performing estimator, achieving an average precision of approximately 0.673 on the test set, while the best-performing estimator, the MLPClassifier, achieved an average precision of approximately 0.889 on the test set.

Based on the average precision scores from these default configurations, we decided to proceed with the provided list of estimators. However, one high-performing estimator, the GaussianProcessClassifier, was excluded from further consideration due to its prohibitively long fitting time.

The final selection included four estimators and a VotingClassifier. These models were chosen based on their strong performance and lack of severe overfitting in their default instantiations. This selection provided a high degree of confidence that we could apply regularization to these models without a significant reduction in performance and further optimize them to enhance their results on our key metrics.

# VI

Model Fine Tuning:

1. RandomForestClassifier
2. LinearSVC
3. RBF_SVC
4. MLPClassifier
5. VotingClassifier

In this section contains the fine tuning procedure of the shortlisted estimators. For all estimators an initial GridSearch is performed over various hyperparameters and the results inspected. Search spaces are then refined for subsequent searches.

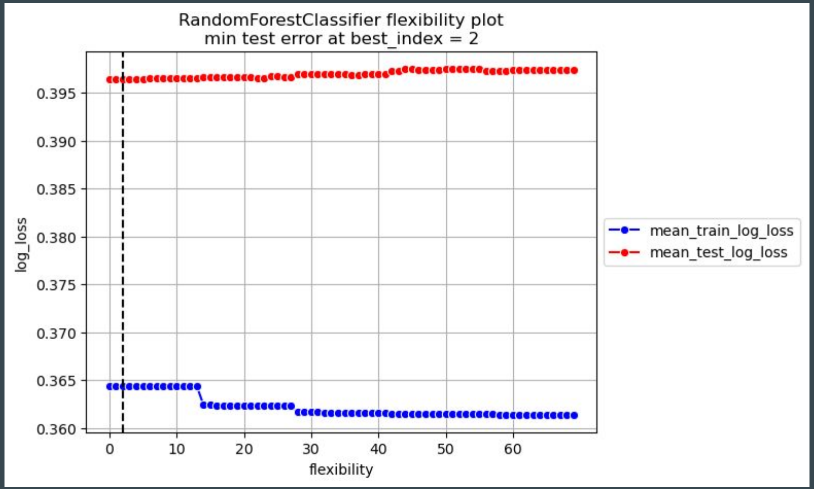
# Model Fine Tuning: RandomForestClassifer

*Hyperparameter Search:*

**n_estimators:**
- The number of trees in the model.

**max_depth:**
- The max depth of the each of the decision trees.

**min_samples_leaf:**
- Minimum number of samples required in a leaf.




Refined Search: Flex Plot

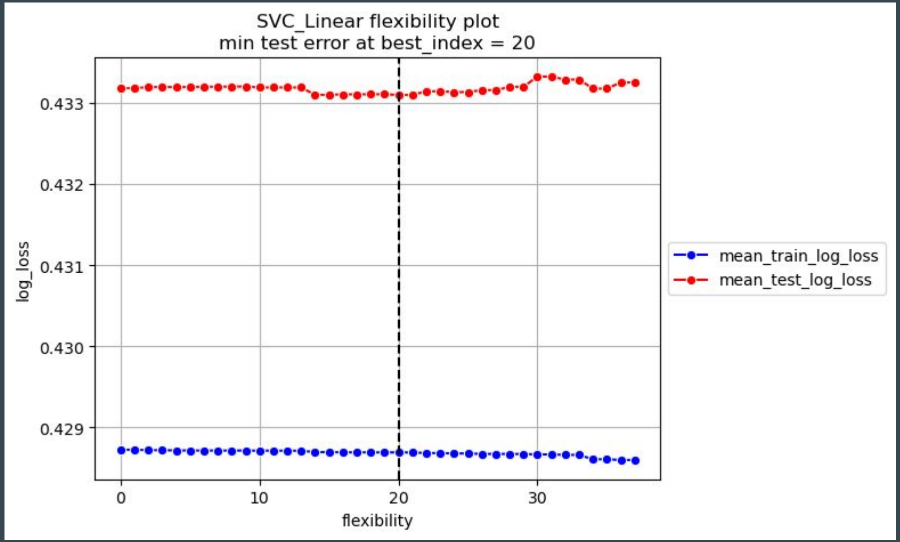
| Hyperparameter | Initial Search Space | Refined Search Space | Best Value |
|---|---|---|---|
| *n_estimators* | `[1, 5, 10, 25, 50, 100, 200, 1000, 1250]` | `[1000, 1050, 1075, 1100, 1150, 1200, 1250]` | `[1200]` |
| *max_depth* | `[2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, None]` | `[7, 8, 9, 10, 11]` | `[7]` |
| *min_samples_leaf* | `list(range(1, 60, 10))` | `None` | `[11]` |

# Model Fine Tuning: LinearSVC

*Hyperparameter Search:*

C:

- The inverse of the of the strength of the regularization.




Refined Search: Flex Plot

| Hyperparameter | Initial Search Space | Refined Search Space | Best Value |
|---|---|---|---|
| *C* | `list(np.arange(0.0, 2.0, step=0.1))` | `list(np.arange(0.7, 1.1, step=0.01))` | [0.89] |

# Model Fine Tuning: SVCRBF

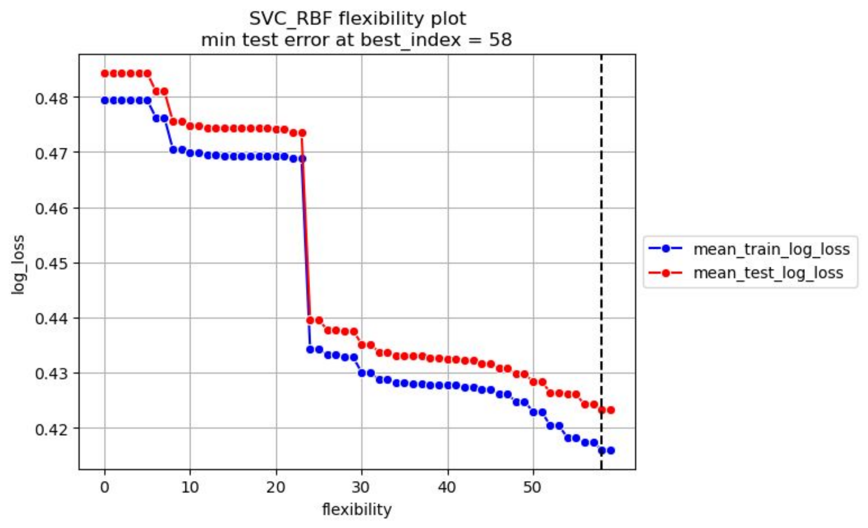
*Hyperparameter Search:*

**C:**

- Trades off correct classification of training examples against maximization of the decision function's margin.

**Gamma:**

- How far the influence of single training example reaches.




Refined Search: Flex Plot

| Hyperparameter | Initial Search Space | Refined Search Space | Best Value |
|---|---|---|---|
| *C* | `list(np.logspace(-3, 3, 10))` | `list(np.logspace(-3, 3, 30))` | `[621.017]` |
| *Gamma* | `['scale', 'auto', 0.01, 0.001]` | `None` | `[.001]` |

# Model Fine Tuning: MLPClassifier

*Hyperparameter Search:*

**hidden_layer_size:**

- The number of neurons in the hidden layer.

**alpha:**

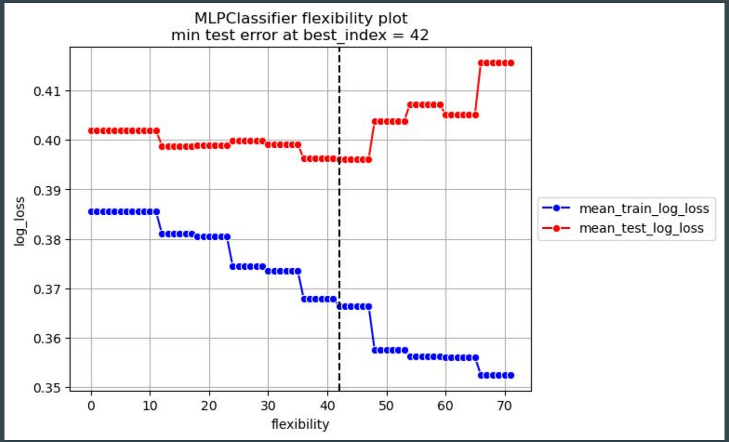
- The strength of regularization, L2.

**Learning_rate:**

- The learning schedule for weight updates, how will the learning rate change..

**Learning_rate_init:**

- The initial learning rate used, controls step-size.

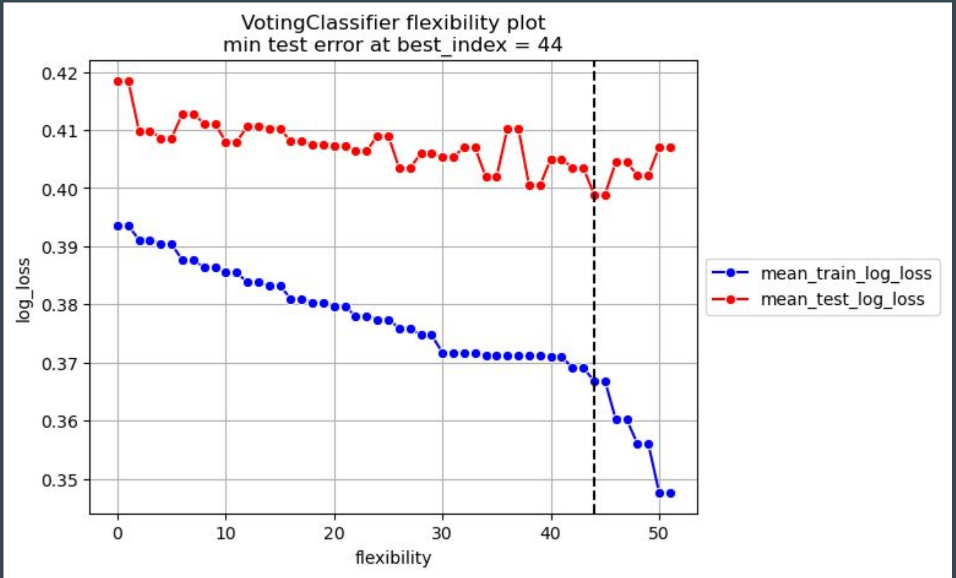



Refined Search: Flex Plot

| Hyperparameter | Initial Search Space | Refined Search Space | Best Value |
|---|---|---|---|
| *hidden_layer_size* | [(50,) (100,) (50, 50)] | [(50,) (100,) (50, 50)] | [(100,)] |
| *alpha* | [0.0001, 0.001, 0.0] | [0.0001, 0.0] | [0.0001] |
| *learning_rate* | ['contant', 'invscaling, 'adaptive'] | ['contant', 'invscaling, 'adaptive'] | ['adaptive'] |
| *learning_rate_init* | [0.001, 0.05, 0.01] | [0.01, 0.001] | [0.01] |

# Model Fine Tuning: Voting Classifier

- After identifying the optimal fit for all the estimators on the shortlist, a gridsearch over various weight combinations of the estimators was performed within the voting classifier.
- The best hypers for the each of the individual models was used in the voting classifier search.




*Hyperparameter Search:*

**estimator_weights:**

- Determines how to weight the prediction of each estimator.

```
# Equal weighting
[1.0, 1.0, 1.0, 1.0],

# Emphasizing individual estimators
[2.0, 1.0, 1.0, 1.0],  # Emphasize SVC_Linear
[1.0, 2.0, 1.0, 1.0],  # Emphasize SVC_RBF
[1.0, 1.0, 2.0, 1.0],  # Emphasize RandomForest
[1.0, 1.0, 1.0, 2.0],  # Emphasize MLPClassifier

# De-emphasizing individual estimators
[0.5, 1.0, 1.0, 1.0],  # De-emphasize SVC_Linear
[1.0, 0.5, 1.0, 1.0],  # De-emphasize SVC_RBF
[1.0, 1.0, 0.5, 1.0],  # De-emphasize RandomForest
[1.0, 1.0, 1.0, 0.5],  # De-emphasize MLPClassifier

# Diverse combinations
[1.0, 1.0, 0.5, 3.0],  # Very high MLPClassifier, lower RandomForest
[1.0, 1.0, 3.0, 0.5],  # Very high RandomForest, lower MLPClassifier
[1.0, 3.0, 1.0, 0.5],  # Very high SVC_RBF, lower MLPClassifier
[3.0, 1.0, 1.0, 0.5],  # Very high SVC_Linear, lower MLPClassifier

# Balancing emphasis
[1.5, 1.0, 1.0, 2.0],  # Higher MLPClassifier and SVC_Linear
[1.0, 1.5, 2.0, 1.0],  # Higher SVC_RBF and RandomForest
[1.0, 2.0, 1.0, 1.5],  # Higher SVC_RBF and MLPClassifier
[1.0, 1.0, 2.5, 1.0],  # Very high RandomForest

# Extreme emphasis on one estimator
[3.0, 0.5, 1.0, 1.0],  # Very high SVC_Linear
[1.0, 3.0, 1.0, 1.0],  # Very high SVC_RBF
[1.0, 1.0, 3.0, 0.5],  # Very high RandomForest
[1.0, 1.0, 0.5, 3.0],  # Very high MLPClassifier

# Gradual variations
[2.0, 2.0, 1.0, 1.0],  # High SVCs, lower others
[1.0, 1.0, 2.0, 2.0],  # High tree-based models
[0.5, 2.0, 2.0, 1.0],  # Higher RandomForest and SVC_RBF
[1.0, 0.5, 1.0, 3.0],  # Very high MLPClassifier, lower SVC_RBF
[1.0, 1.0, 0.5, 2.5],  # High MLPClassifier, lower RandomForest
[2.0, 0.5, 1.0, 2.0],  # Emphasize MLP and SVC_Linear, lower SVC_RBF
```

**Best Weights: [1.0, 1.0, 0.5, 3.0]** *Very high MLPClassifier, Lower on RandomForest*

# Final Search & Model Promotion

Having discovered the best hyperparameter combinations for all the shortlisted models and the voting classifier, all the models were compared in effort to select the most performant.

The figures on the right display the ranking metric results 10-fold cross validation.

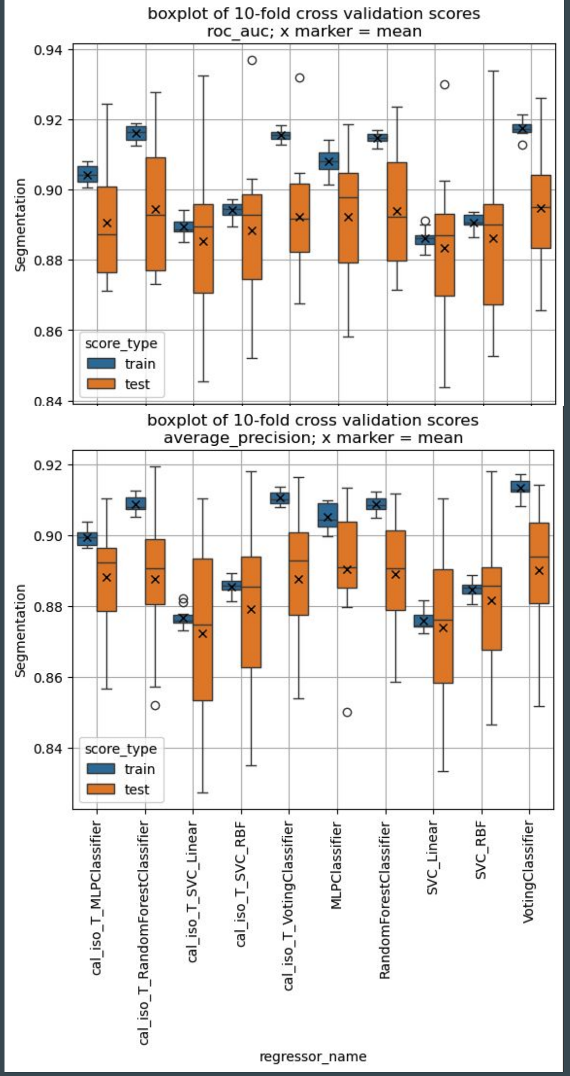
All models underwent probability calibration.

All the models appear reasonable well fit across both metrics, AUC ROC and Average Precision.

Based on Average Precision, Log Loss, & ROC AUC:

- *The model that will be promoted to the test set is:* *VotingClassifier*

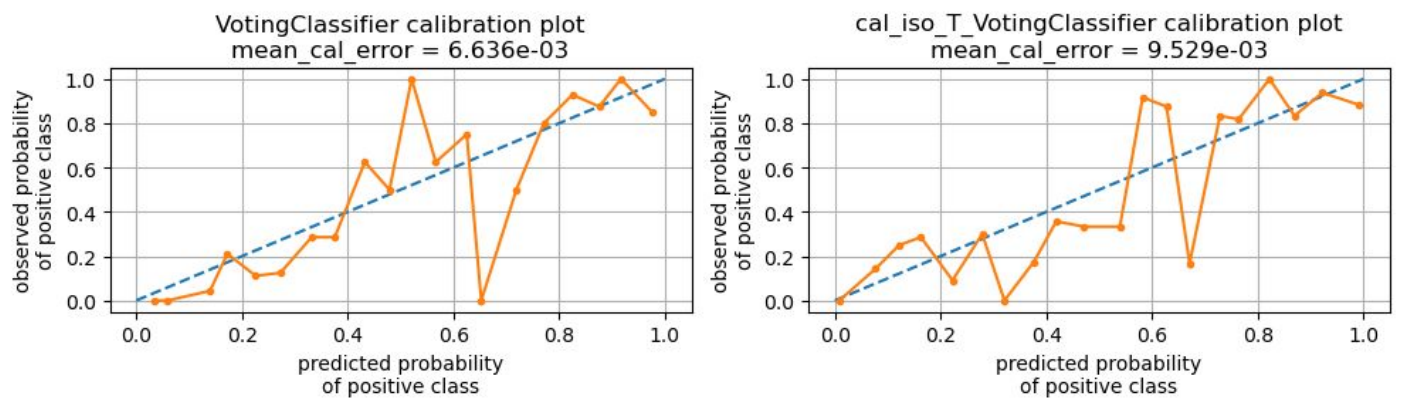
| Estimator | Average Precision | ROC AUC |
|---|---|---|
| MPLClassifier | 0.890458 | 0.892320 |
| RandomForestClassifier | 0.889133 | 0.894026 |
| SVCLinear | 0.874035 | 0.883342 |
| SVCRBF | 0.881513 | 0.886098 |
| VotingClassifier | 0.890217 | 0.894761 |
| cal_iso_MLPClassifier | 0.888155 | 0.890622 |
| cal_iso_RandomForestClassifier | 0.887733 | 0.894503 |
| cal_iso_SVC_Linear | 0.872395 | 0.885235 |
| cal_iso_SVC_RBF | 0.879270 | 0.892145 |
| cal_iso_Voting_Classifier | 0.887757 | 0.892145 |



# Calibration of Promoted Model: VotingClassifier

Calibrating probabilities is an important step when using classifiers in SkLearn, as raw outputs are not always perfectly aligned with true probabilities. Since the VotingClassifier is an ensemble based methods, it naive calibration is based the calibration of  each classifier in the voting ensemble.  Some models might have a canonical link, for example LogisticRegression, however none of our shortlisted models have a canonical link function.

Fine-tuning with isotonic regression, via the *CalibratedClassifierCV* wrapper, can further improve the alignment between predicted probabilities and actual outcomes. Isotonic regression ensures monotonicity and provides more accurate probability estimates by fitting a non-decreasing function to the predicted probabilities. An alternative strategy is to calibrate each model in the ensemble, however we did not attempt this method.




| regressor_name | score_name | score_type | score |
|---|---|---|---|
| VotingClassifier | average_precision | test | 0.890814 |
| VotingClassifier | average_precision | train | 0.913187 |
| VotingClassifier | roc_auc | test | 0.895157 |
| VotingClassifier | roc_auc | train | 0.917263 |

| regressor_name | score_name | score_type | score |
|---|---|---|---|
| cal_iso_T_VotingClassifier | average_precision | test | 0.886711 |
| cal_iso_T_VotingClassifier | average_precision | train | 0.910419 |
| cal_iso_T_VotingClassifier | roc_auc | test | 0.892017 |
| cal_iso_T_VotingClassifier | roc_auc | train | 0.915637 |

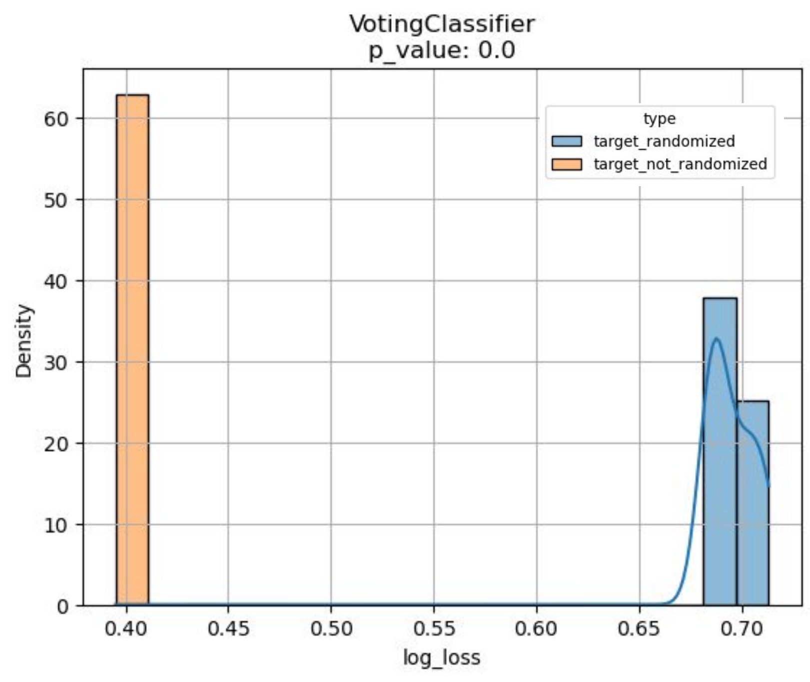
# Checking For False Discoveries

The histograms display the Log Loss  distributions for both target_randomized and target_not_randomized datasets.

The target_randomized results form  the null distribution, representing the Log Loss values of models trained on data with no predictive relationship. These values represent what the Log Loss  would look like if the features had were randomly aligned with the target.

The target_not_randomized values shows the Log Loss  of models trained on the original data, where the relationship between the features and target are preserved. These errors are noticeably smaller compared to the randomized data, and the distribution is more concentrated. This indicates that there is a stronger predictive relationship between the feature and the target.

If the non-randomized  Log Loss  values had fallen within the range of the null distribution, it would suggest that the model has no better predictive power than a random guess, indicating a potential false discovery - since the null distribution indicates the results expected under the conditions of  the null hypothesis. Values far outside the range of the null distribution, as seen here,  allow use to reject the null hypothesis.




This figure demonstrates we    NOT made a false discovery.

# VII

1. Key Findings
2. Summation and Review of Process and Methods
3. Conclusion

# Key Findings

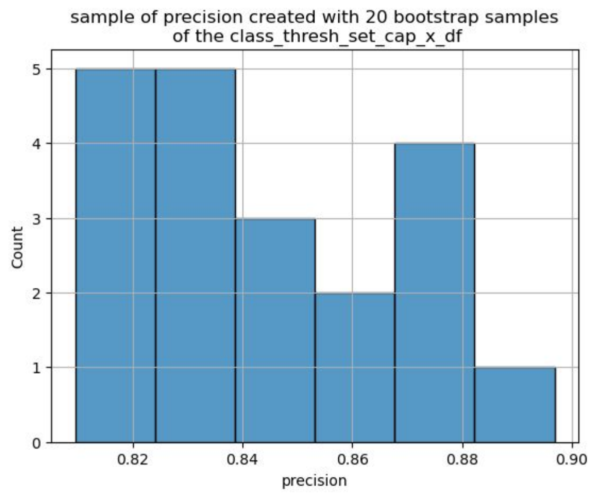
# Results on Test Set

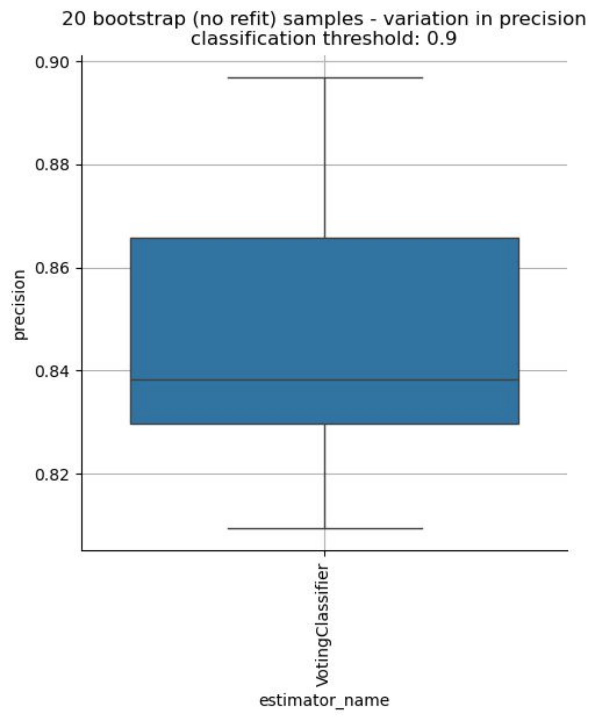
**Evaluation of Key Metric: Precision**

**Min:** 0.8095238095238095
**Mean:** 0.8448487960711815
**Max:** 0.8968253968253969

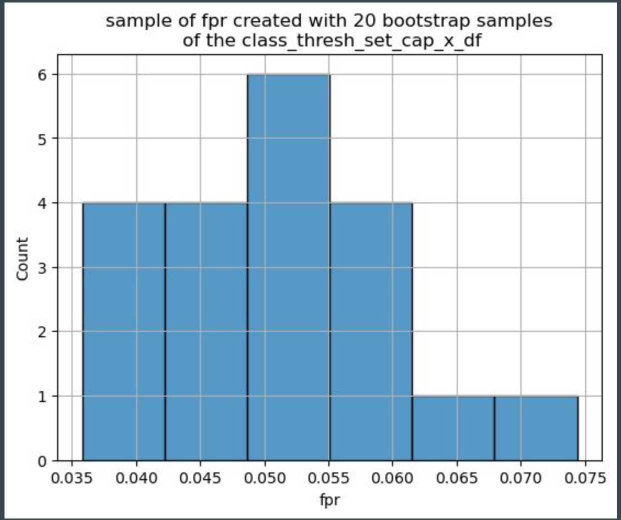
Percentile based 95% confidence interval ranges from:
0.8111 to 0.8772

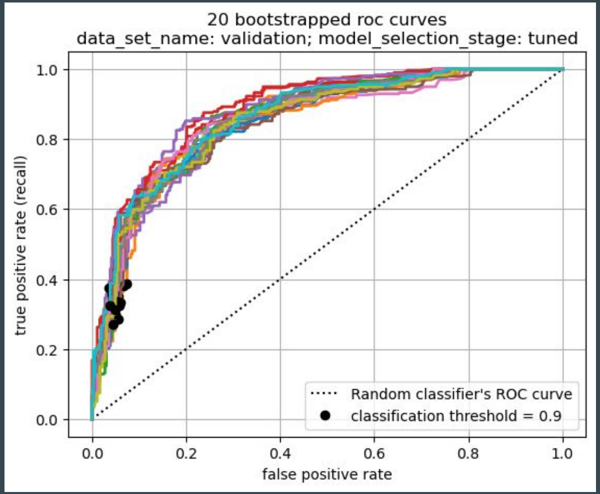
These results reflect a strong and consistent performance in terms of precision, with limited variability. The narrow confidence interval further supports the reliability of the model in maintaining precision above the target threshold of 80% - indicating success based on our initial proposal.



# More Results



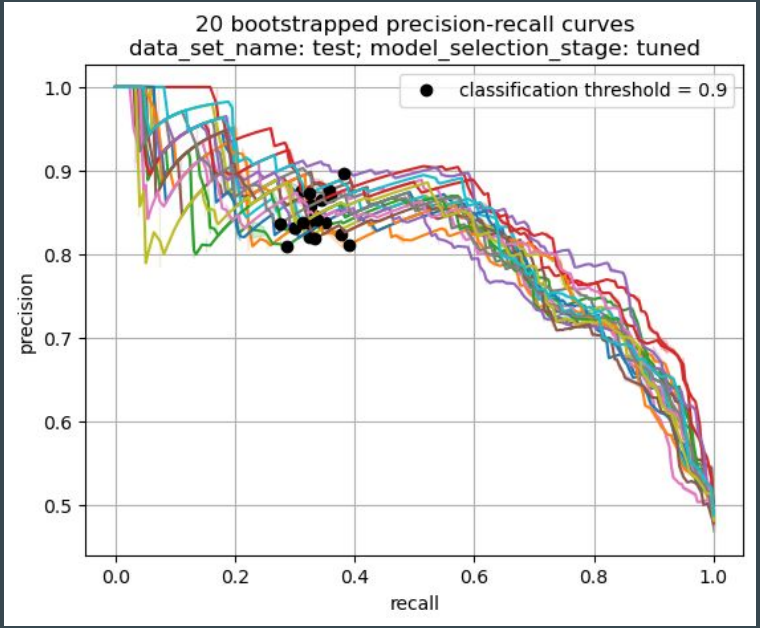

**False Positive Rate:**
Min: 0.03581267217630854
Mean: 0.05
Max: 0.0743801652892562
Percentile based 95% confidence interval ranges from:
0.0358 to 0.0665

**ROC-AUC:**
Min: 0.8401879214551391
Mean: 0.8574723358469916
Max: 0.8876645240281604
Percentile based 95% confidence interval ranges from:
0.8418 to 0.8824

# Interpretation

The precision-recall curve for 20 bootstrapped samples from the test set illustrates the trade-off between precision and recall for our tuned and calibrated classifier. There is a notable trade-off between these metrics, with the selected classification threshold reflecting our decision to prioritize precision. Increasing the recall score beyond 60% would result in an unacceptable reduction in precision, falling below our target of 80%.

Although the curve is not ideal—an ideal curve would show minimal trade-offs—it demonstrates relative stability in precision values. This stability provides confidence in the reliability of our performance metrics and the robustness of the model under the chosen threshold.



# Review of Methods

# Threshold Tuning

**Classification Threshold = 0.9**

| class | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.6503 | 0.9657 | 0.7772 | 233 |
| 1.0 | 0.8961 | 0.3632 | 0.5169 | 190 |

Expected

Predicted

| Class | B | ~B |
|---|---|---|
| B | TP = 225 | FP = 8 |
| ~B | FN = 121 | TN = 69 |

B (class 1) vs D (class 0) Confusion Matrix

Class 0 (D):

- High recall (~0.966): Most of the actual class 0 instances are correctly identified.
- Moderate precision (~0.650): Many predicted class 0 instances are correct, but a notable proportion are false positives.

Class 1 (B):

- High precision (~0.896): Most predicted class 1 instances are correct.
- Low recall (~0.363): A significant portion of actual class 1 instances are missed.

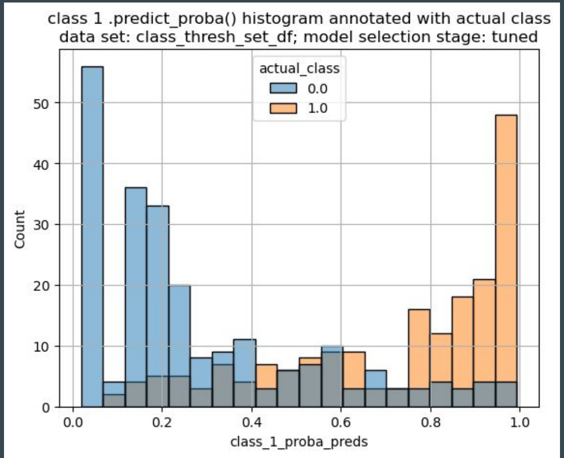
This demonstrates that the model prioritizes reducing mislabeling of class 0 as class 1 - which aligns with the stated goal of minimizing false positives for class 1.

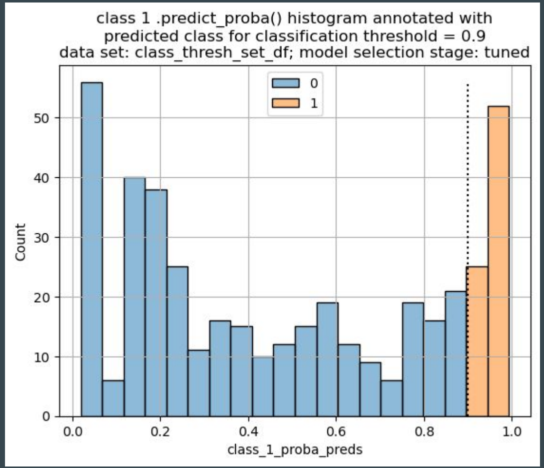
# Threshold Tuning (continued)

We selected a classification threshold of 0.9 to minimize the risk of false positives for class 1. This decision was driven by the high cost associated with physical mailers and the belief that individuals misclassified as class 0 could still be reached through alternative, albeit slightly less effective, means.

The primary goal behind this threshold adjustment was to reduce wasted spending on physical mailers by ensuring a high precision for class 1. By prioritizing precision, we aimed to target individuals who are highly likely to belong to class 1, thereby maximizing the return on investment for the mailer campaign. This approach acknowledges the trade-off between precision and recall, accepting a potential decrease in recall (fewer true class 1 individuals identified) in favor of greater confidence in the predictions for class 1.

Ultimately, this strategy reflects a cost-sensitive classification framework, where the financial implications of a false positive outweigh the opportunity cost of a false negative. By setting a stringent threshold, we aligned the model's predictions with the campaign's economic and operational priorities.



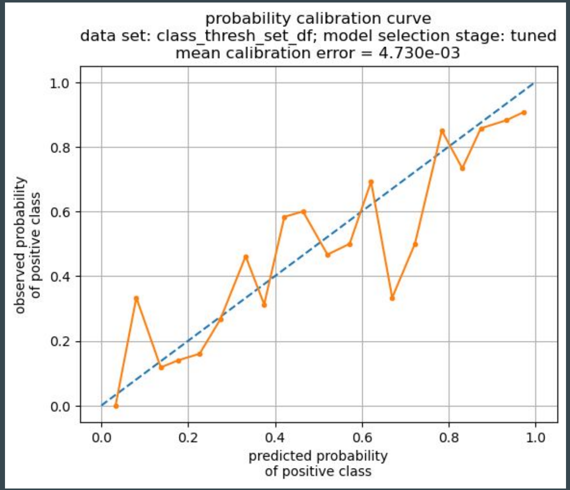
# Final Calibration

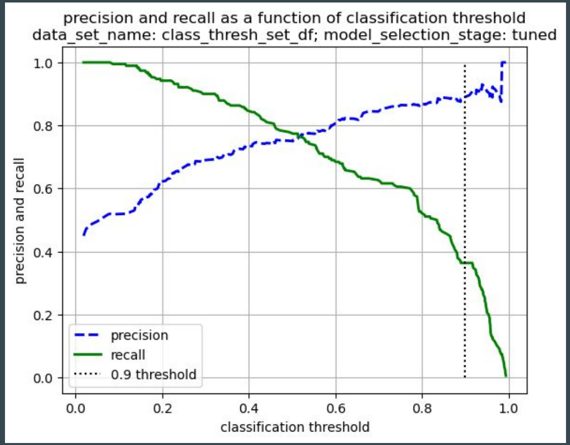
The calibration curve for the promoted model, displayed in orange, is a fairly close match to the ideal curve, displayed with the dotted blue line.

This indicates that the model's predicted probabilities are well-aligned with the actual observed probabilities.

This suggests that the model's probability estimates are reliable, meaning the confidence scores it assigns to predictions accurately reflect the likelihood of correctness.

The precision recall curves, as a function of the calibration some some minor instability due to the irregularities in the calibration curve. However, the general shape of the trade-off is preserved and not significantly impacted by the calibration, indication the models robustness to calibration.



# Conclusion

# Assumptions & Limitations

**Assumptions** :

- Our key assumption was that we could deliver results more efficiently and with less cost to the stakeholder by taking what was initially a multi classification problem and creating a binary classification scheme that attempted to solve the same problem.
- This decision made due largely for practical reasons, as we wanted to utilize the knowledge learned and the flows developed over the course of the semester, rather than tackle multiclass classification. To this end, we developed logical framing for this choice that reflected how a decision to pursue binary over multiclass might play out in our project's setting. Due to the actual costs in compute and labor involved with multiclass, this seemed to be a realistic choice.

**Limitations** :

- Our model is limited in its performance in that as precision is tuned to an acceptable level, recall decreases dramatically.
- This has not prevented us from reaching the business objective, but is nevertheless not an ideal outcome.
- Ideally, we would like to be able to tune precision without decreasing recall.

# Finals Thoughts

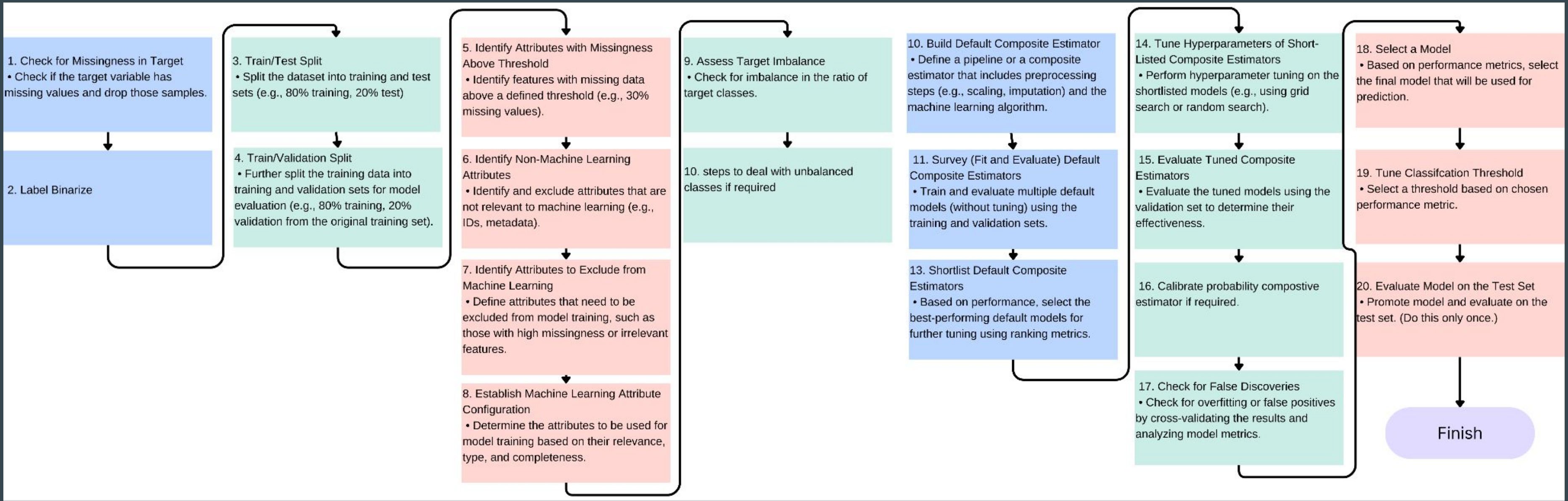
- Mean test precision on the test set for the positive class is: **85.4%,**
- The 95% confidence interval of mean test precision is: **81.1% to 87.7%**
- The minimum test precision seen in bootstrapping is: **81%**
- This demonstrates the model is generalizing well to unseen data.
- All of these values exceed the minimum threshold set for the project to be considered a success, **80.0%** .
- The model can be reliably put into production.

# Acknowledgements

1. Steven Morin PhD., DS 5220 Class Materials
2. Sklearn.org

# Thank you!

# Appendix


1. Check for Missingness in Target
• Check if the target variable has missing values and drop those samples.
2. Label Binarize
3. Train/Test Split
• Split the dataset into training and test sets (e.g., 80% training, 20% test)
4. Train/Validation Split
• Further split the training data into training and validation sets for model evaluation (e.g., 80% training, 20% validation from the original training set).
5. Identify Attributes with Missingness Above Threshold
• Identify features with missing data above a defined threshold (e.g., 30% missing values).
6. Identify Non-Machine Learning Attributes
• Identify and exclude attributes that are not relevant to machine learning (e.g., IDs, metadata).
7. Identify Attributes to Exclude from Machine Learning
• Define attributes that need to be excluded from model training, such as those with high missingness or irrelevant features.
8. Establish Machine Learning Attribute Configuration
• Determine the attributes to be used for model training based on their relevance, type, and completeness.
9. Assess Target Imbalance
• Check for imbalance in the ratio of target classes.
10. steps to deal with unbalanced classes if required
10. Build Default Composite Estimator
• Define a pipeline or a composite estimator that includes preprocessing steps (e.g., scaling, imputation) and the machine learning algorithm.
11. Survey (Fit and Evaluate) Default Composite Estimators
• Train and evaluate multiple default models (without tuning) using the training and validation sets.
13. Shortlist Default Composite Estimators
• Based on performance, select the best-performing default models for further tuning using ranking metrics.
14. Tune Hyperparameters of Short-Listed Composite Estimators
• Perform hyperparameter tuning on the shortlisted models (e.g., using grid search or random search).
15. Evaluate Tuned Composite Estimators
• Evaluate the tuned models using the validation set to determine their effectiveness.
16. Calibrate probability compostive estimator if required.
17. Check for False Discoveries
• Check for overfitting or false positives by cross-validating the results and analyzing model metrics.
18. Select a Model
• Based on performance metrics, select the final model that will be used for prediction.
19. Tune Classifcation Threshold
• Select a threshold based on chosen performance metric.
20. Evaluate Model on the Test Set
• Promote model and evaluate on the test set. (Do this only once.)
Finish