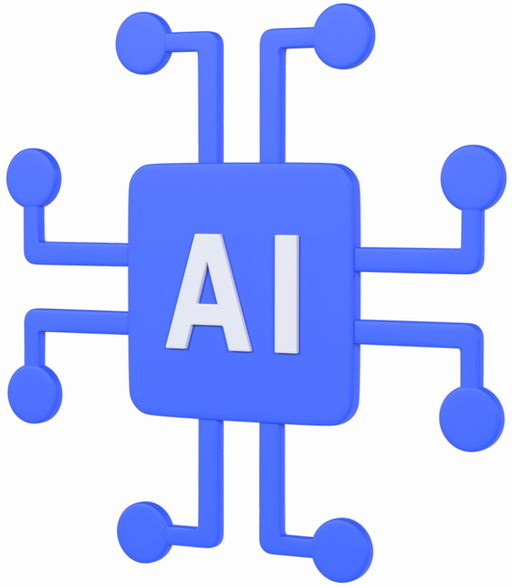



# PROJECT MACHINE LEARNING

CLÉMENCE, GEMMA, JONATHON

# TABLE OF CONTENTS

# EDA AND DATA CLEANING, DESCRIPTIVE ANALYTICS

- Check for missing data
- Check feature types
- Clean feature headers
- Check for high collinearity
- Check for low variance
- Balance data using SMOTE

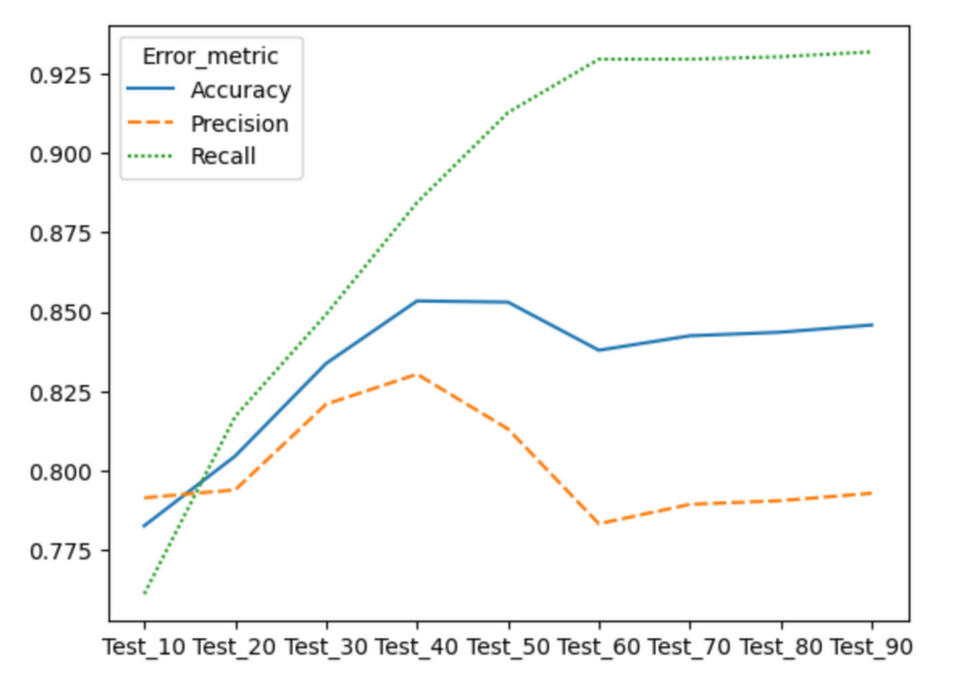
# BERNOULLI NB


# BERNOULLI NB

- BernoulliNB is a type of Naive Bayes classifier that is particularly well-suited for binary or multiclass classification problems with sparse data. It models the probability of each class given the presence or absence of each feature, using the Bernoulli distribution.

- BernoulliNB does not have a hierarchy of rules or splits like decision trees or other tree-based models do. Instead, it directly models the joint probability of the input features and the class labels.

- It makes strong independence assumptions between the features, which can make it less effective for complex problems where the features are interdependent.



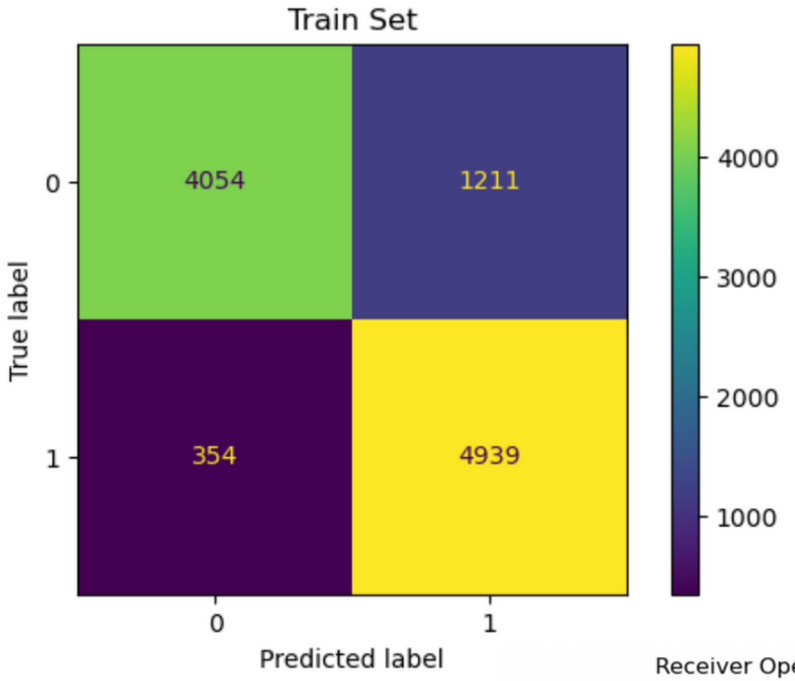

All features

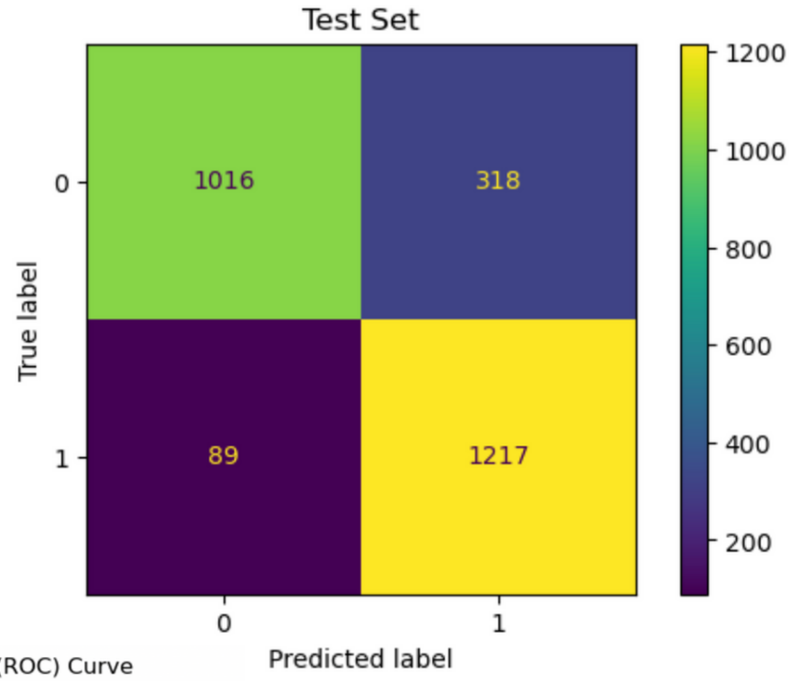
| | Error_metric | Train | Test |
|---|---|---|---|
| **0** | Accuracy | 0.83 | 0.82 |
| **1** | Precision | 0.77 | 0.76 |
| **2** | Recall | 0.93 | 0.92 |

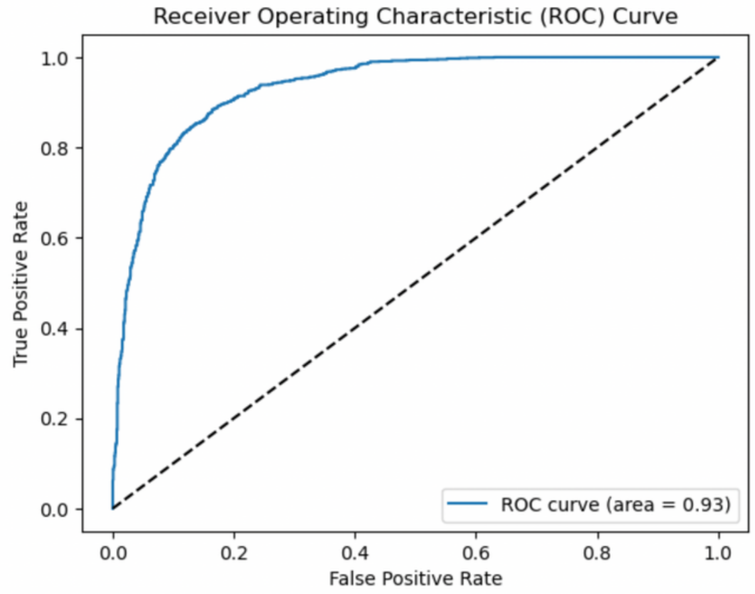
With 50 features

| | Error_metric | Train | Test |
|---|---|---|---|
| **0** | Accuracy | 0.86 | 0.85 |
| **1** | Precision | 0.82 | 0.81 |
| **2** | Recall | 0.91 | 0.91 |

# BERNOULLI NB



# ADABOOS CLASSIFIER

# ADABOOST(ADAPTIVE BOOSTING) CLASSIFIER

"Adaboost iteratively trains a series of base classifiers on a weighted subset of the training data."

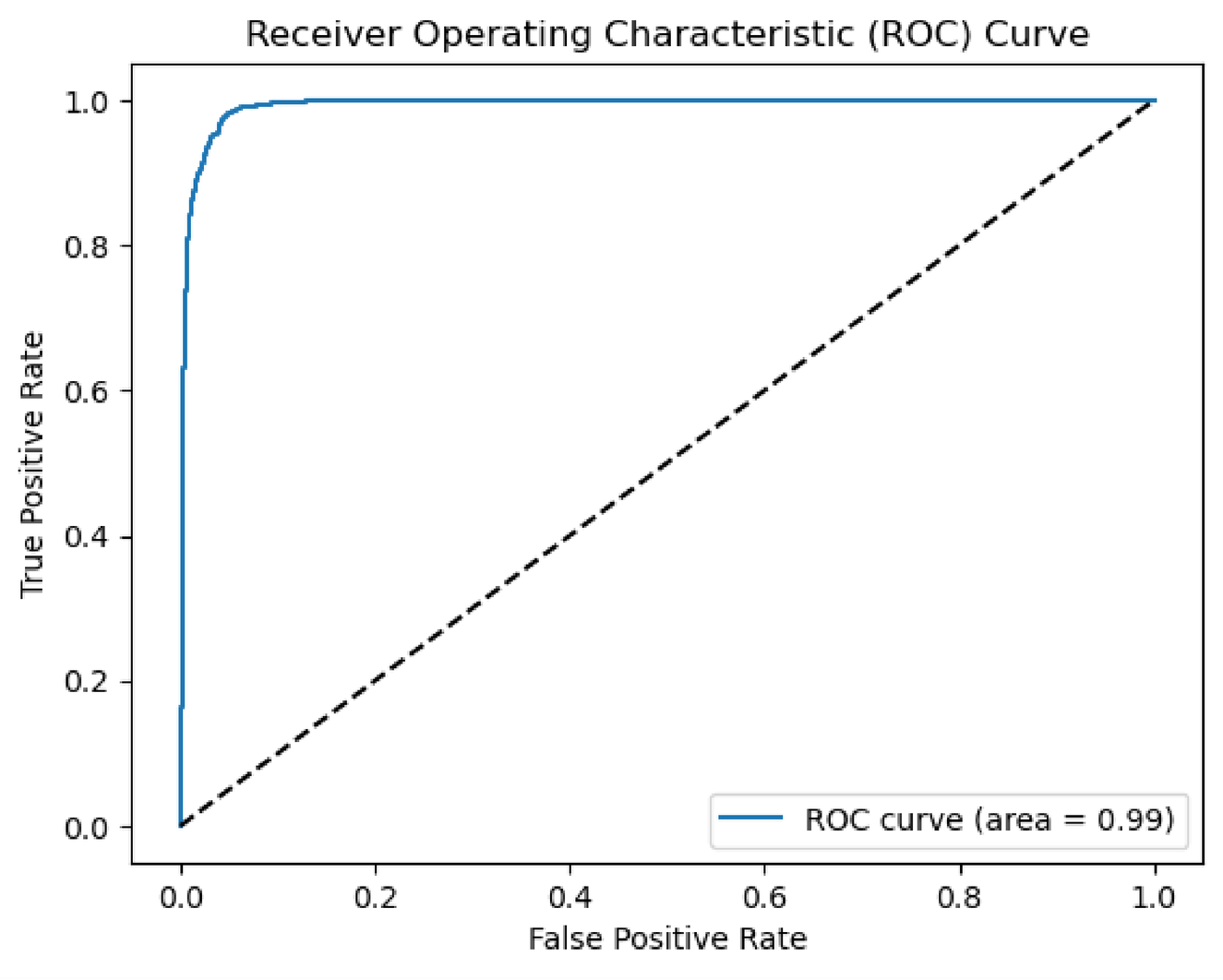
- It iterates through a base classifier and assesses how accurate each iteration is based on the total error.

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

- It then runs additional models based on a weighted subset of this data.

- The final outcome compares the sums of "amount of say" for models that correctly predict the dependent variable vs models that don't.

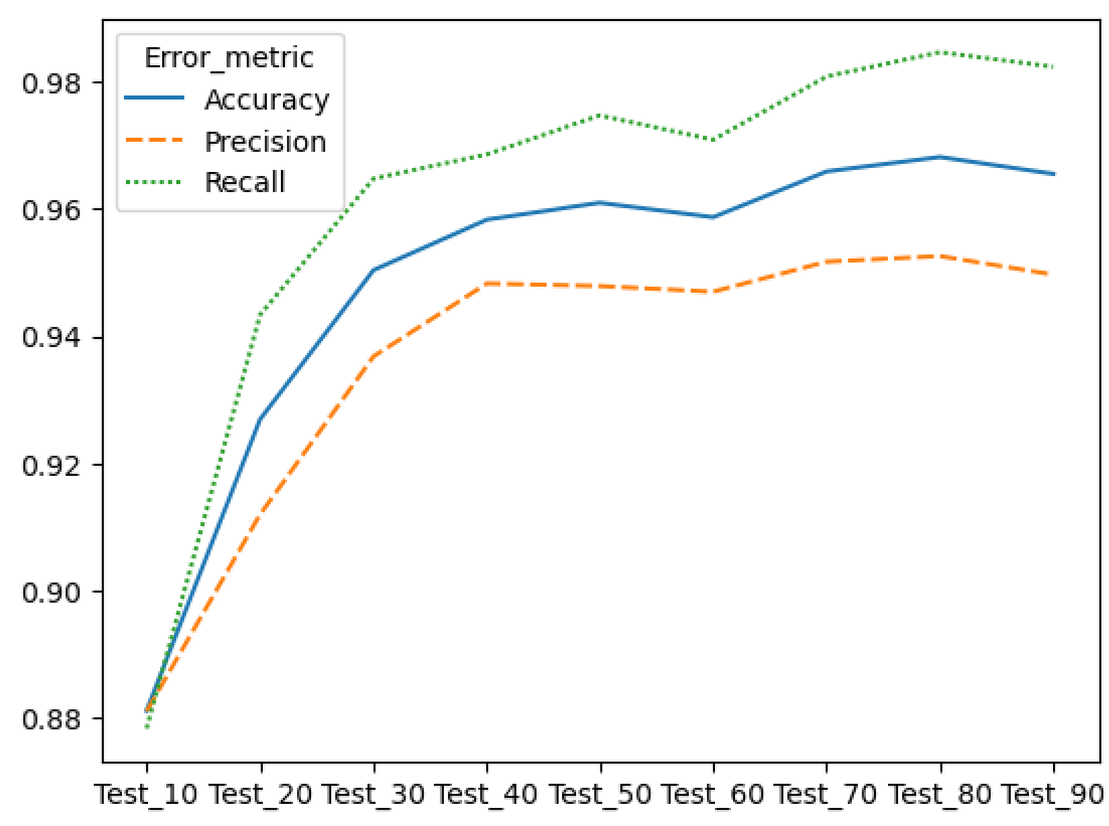



AdaBase using all features

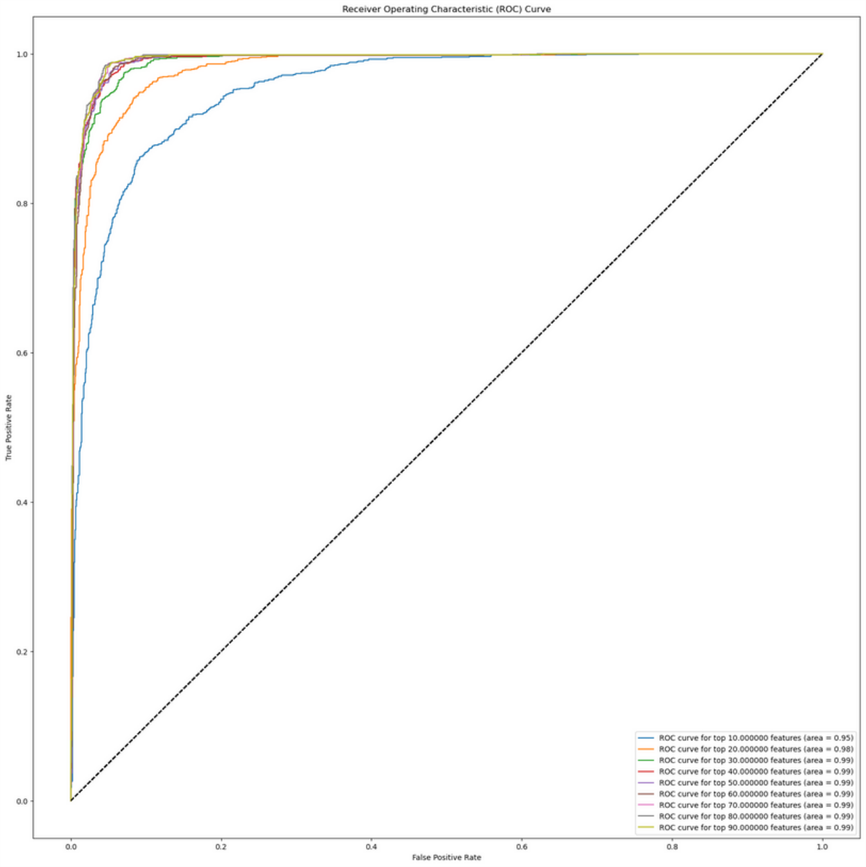
| | Error_metric | Train | Test |
|---|---|---|---|
| 0 | Accuracy | 0.980205 | 0.965530 |
| 1 | Precision | 0.969350 | 0.949002 |
| 2 | Recall | 0.991876 | 0.983155 |

```
tree = DecisionTreeClassifier(criterion='gini', max_depth=1, random_state=42)
tree = tree.fit(X_train_scaled, y_train)

ada = AdaBoostClassifier(base_estimator=tree, n_estimators=500, learning_rate=0.5, random_state=42)
ada.fit(X_train_scaled, y_train)
```

## ACCURACY, PRECISION AND RECALL FOR DIFFERENT NUMBERS OF FEATURES



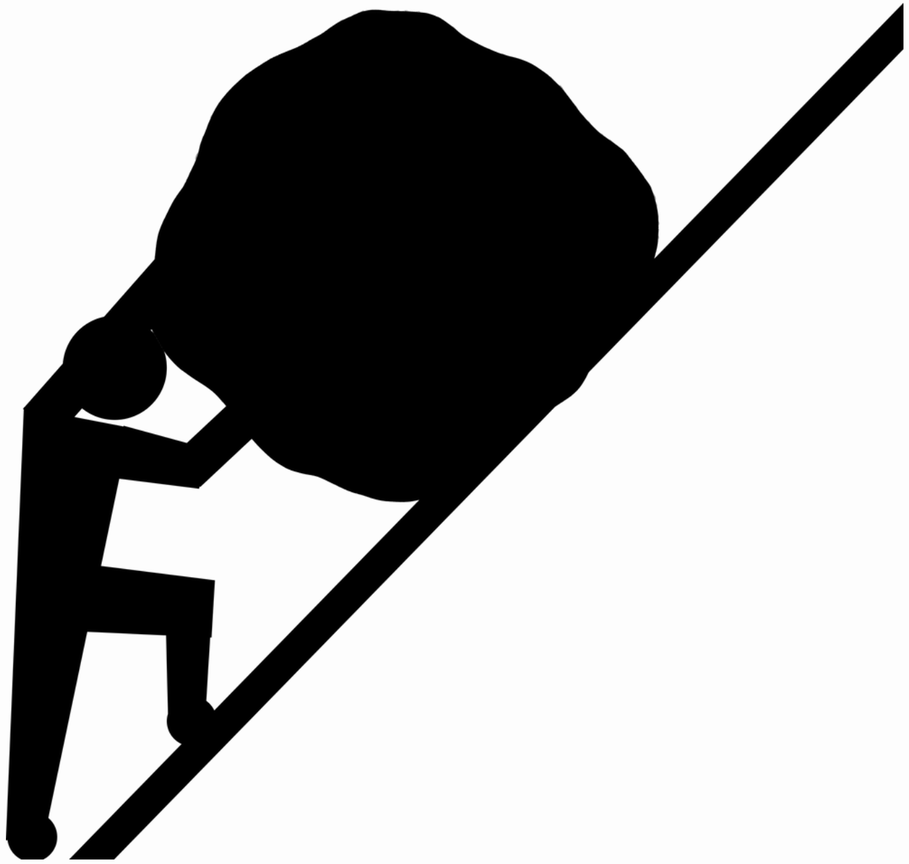

## ROC CURVES FOR DIFFERENT NUMBERS OF FEATURES




| | Error_metric | Test_10 | Test_20 | Test_30 | Test_40 | Test_50 | Test_60 | Test_70 | Test_80 | Test_90 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Accuracy | 0.881061 | 0.926894 | 0.950379 | 0.958333 | 0.960985 | 0.958712 | 0.965909 | 0.968182 | 0.965530 |
| 1 | Precision | 0.880952 | 0.911917 | 0.936803 | 0.948276 | 0.947878 | 0.946975 | 0.951709 | 0.952593 | 0.949667 |
| 2 | Recall | 0.878254 | 0.943338 | 0.964778 | 0.968606 | 0.974732 | 0.970904 | 0.980858 | 0.984686 | 0.982389 |

# HISTGRADIENTBOOSTING CLASSIFIER

# HISTGRADIENTBOOSTING CLASSIFIER



- HistGradientBoostingClassifier is a machine learning algorithm used for classification tasks. It belongs to the family of boosting algorithms. Boosting refers to a class of ensemble learning algorithms that add tree models to an ensemble sequentially.

- The term "hist" in HistGradientBoostingClassifier refers to that instead of computing gradients for every single instance in the training data, the algorithm builds histograms of the features to group similar instances together.

# HISTGRADIENTBOOSTING CLASSIFIER

```
from sklearn.model_selection import GridSearchCV
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier

param_grid = {
    'learning_rate': [0.1, 0.05, 0.01],
    'max_depth': [3, 5, 7],
    'max_iter': [100, 200, 300]
}

hgb = HistGradientBoostingClassifier()

grid_search = GridSearchCV(hgb, param_grid, cv=5)

grid_search.fit(X_train_scaled, y_train)

best_params = grid_search.best_params_
```

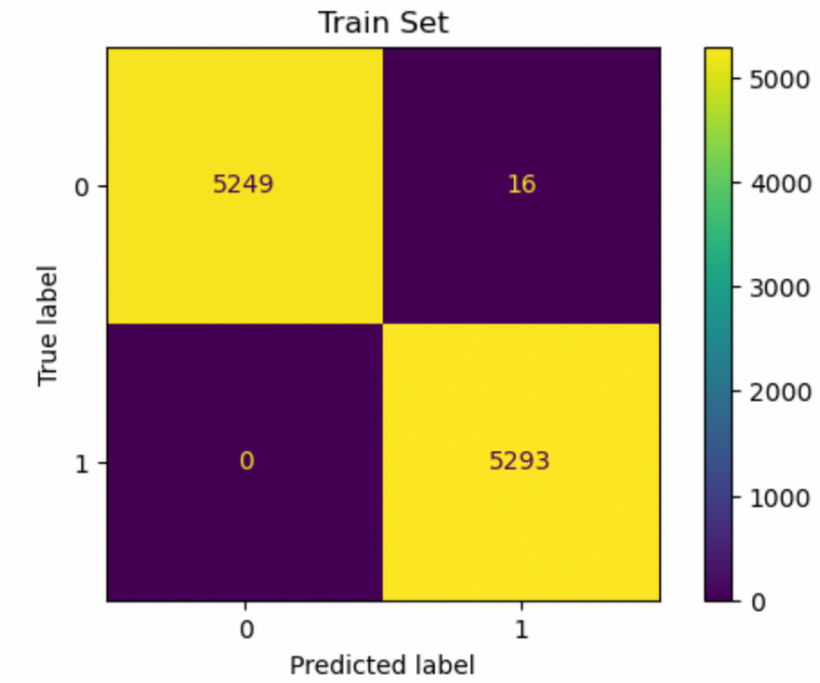
```
best_params
```

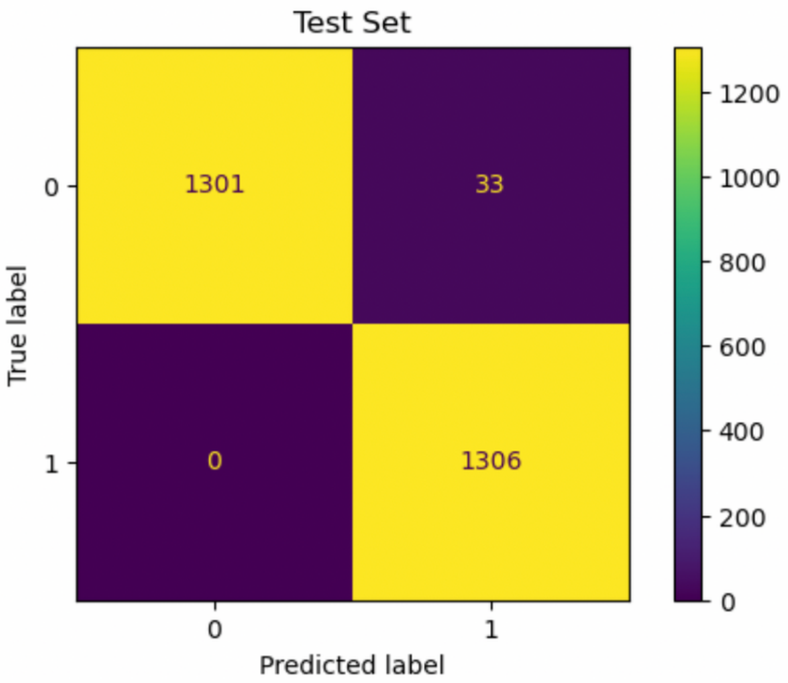
```
{'learning_rate': 0.1, 'max_depth': 7, 'max_iter': 300}
```

```
hgb = HistGradientBoostingClassifier(**best_params)
hgb.fit(X_train_scaled, y_train)
y_pred_test = bnb.predict(X_test_scaled)
y_pred_train=bnb.predict(X_train_scaled)
```

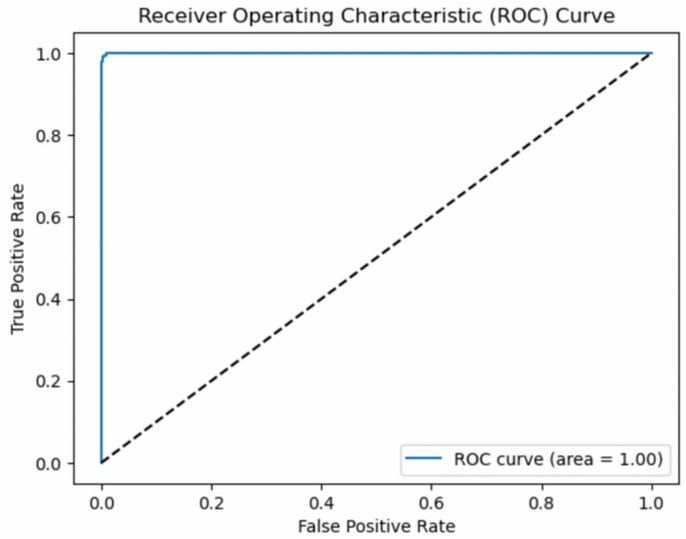
| | Error_metric | Train | Test |
|---|---|---|---|
| 0 | Accuracy | 0.998 | 0.988 |
| 1 | Precision | 0.997 | 0.975 |
| 2 | Recall | 1.000 | 1.000 |

# HISTGRADIENTBOOSTING CLASSIFIER

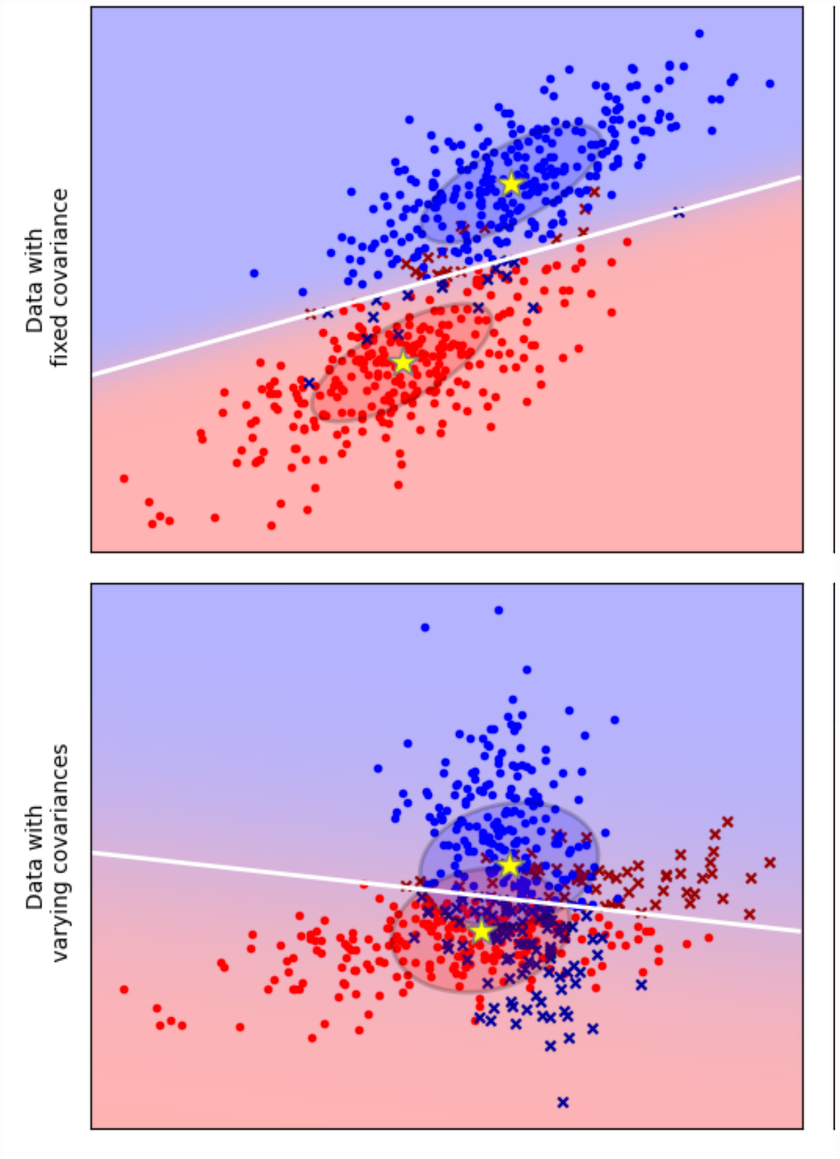



| | Error_metric | Train | Test |
|---|---|---|---|
| **0** | Accuracy | 0.998 | 0.988 |
| **1** | Precision | 0.997 | 0.975 |
| **2** | Recall | 1.000 | 1.000 |



# LINEAR DISCRIMINATION ANALYSIS




# LINEAR DISCRIMINATION ANALYSIS

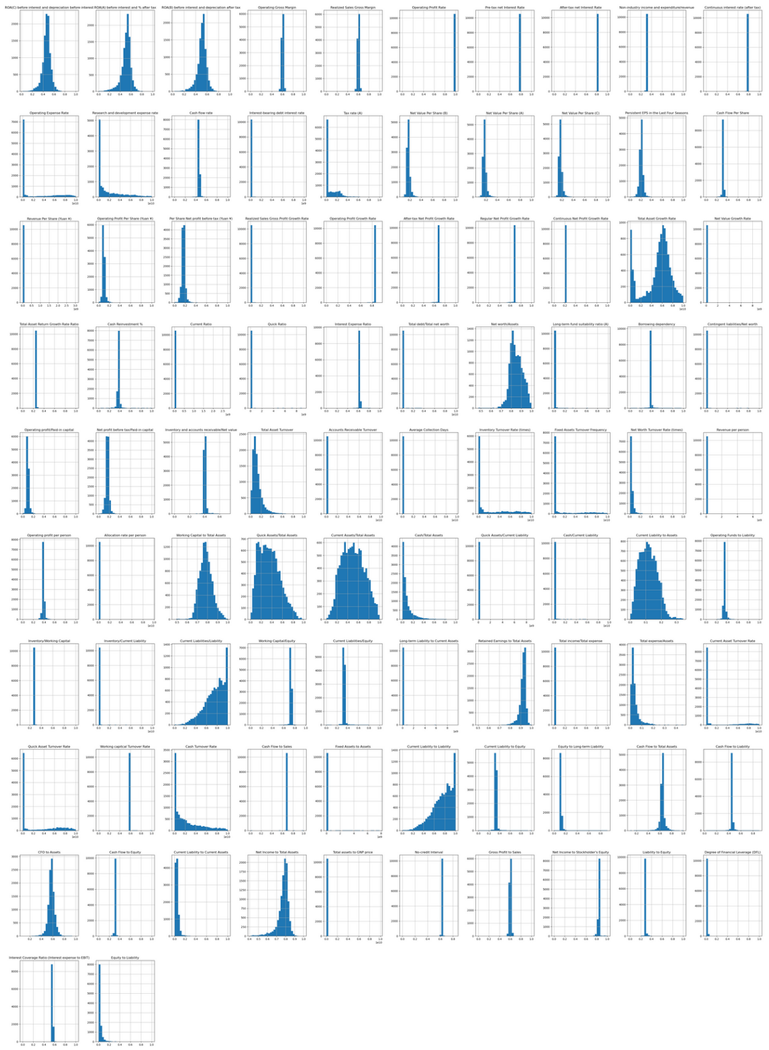
Statistical technique used for classifying data into distinct groups or categories based on a set of predictor variables.

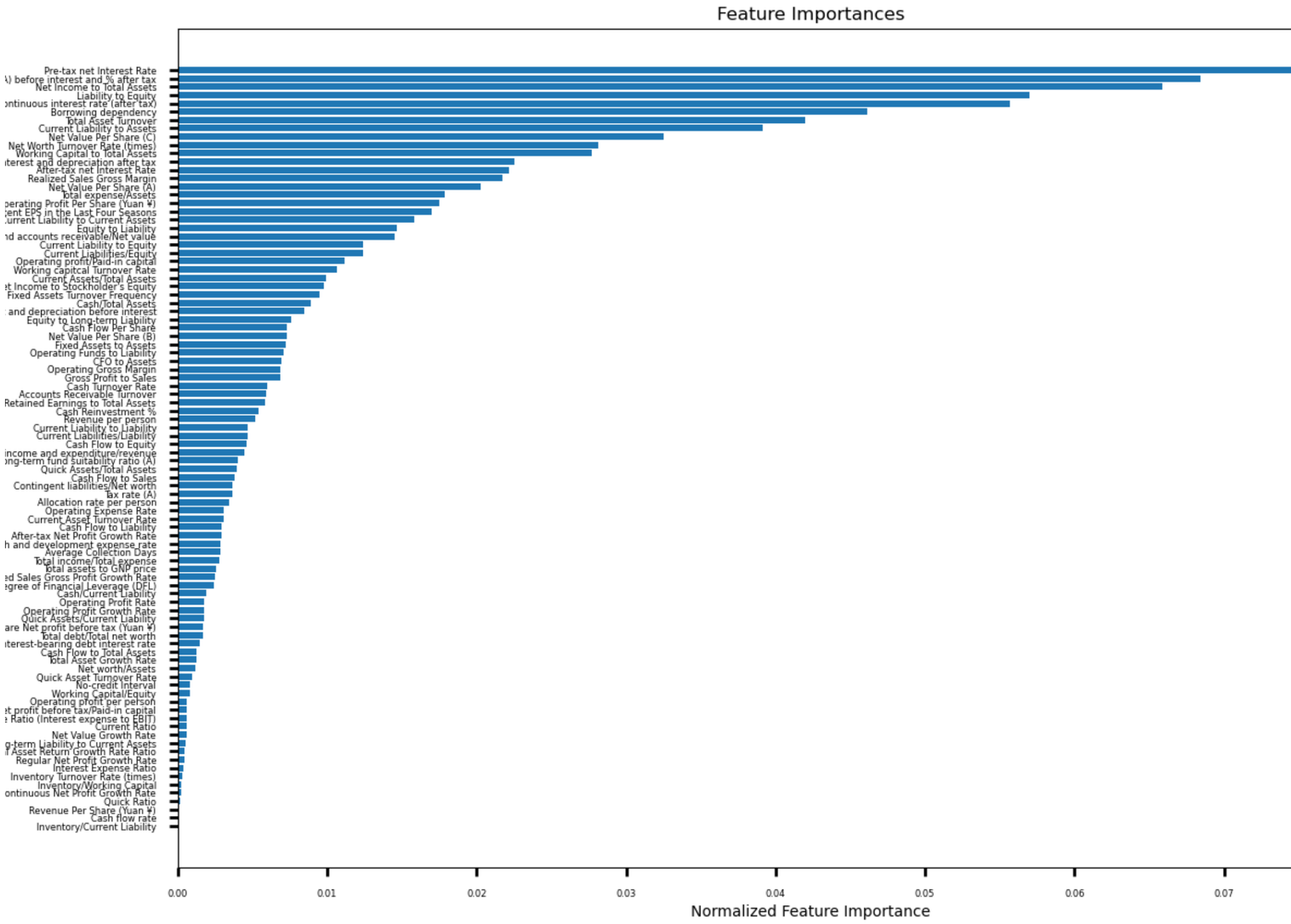
Assumes that:

- predictor variables are normally distributed
- have equal variances within each group.

The goal:
find a linear combination of the predictor variables that maximally separates the groups while minimizing the variation within each group.

# Power Transform & feature importance





# HYPER-PARAMETER TUNING

**01** Possible to tune parameter, but unneccesary in this scenario

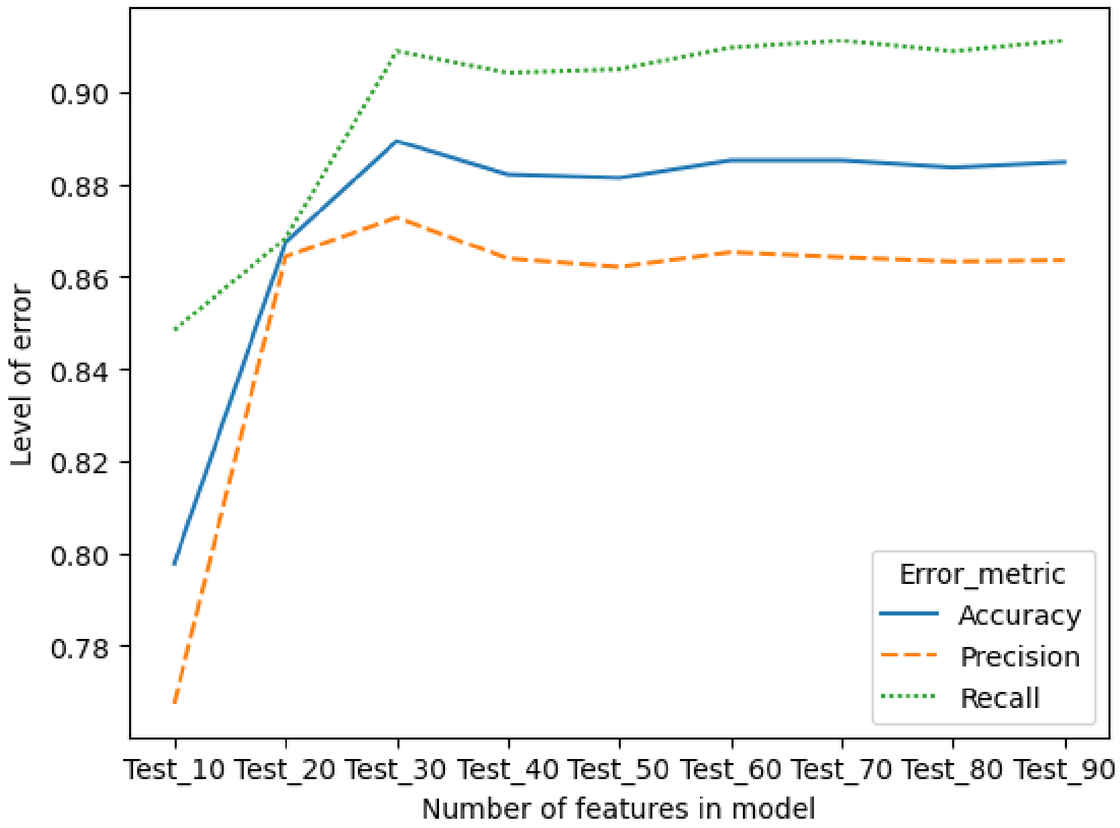
**02** Gridsearch returned multiple good models, and didnt have a 'best_params_

**02** Some key parameters:

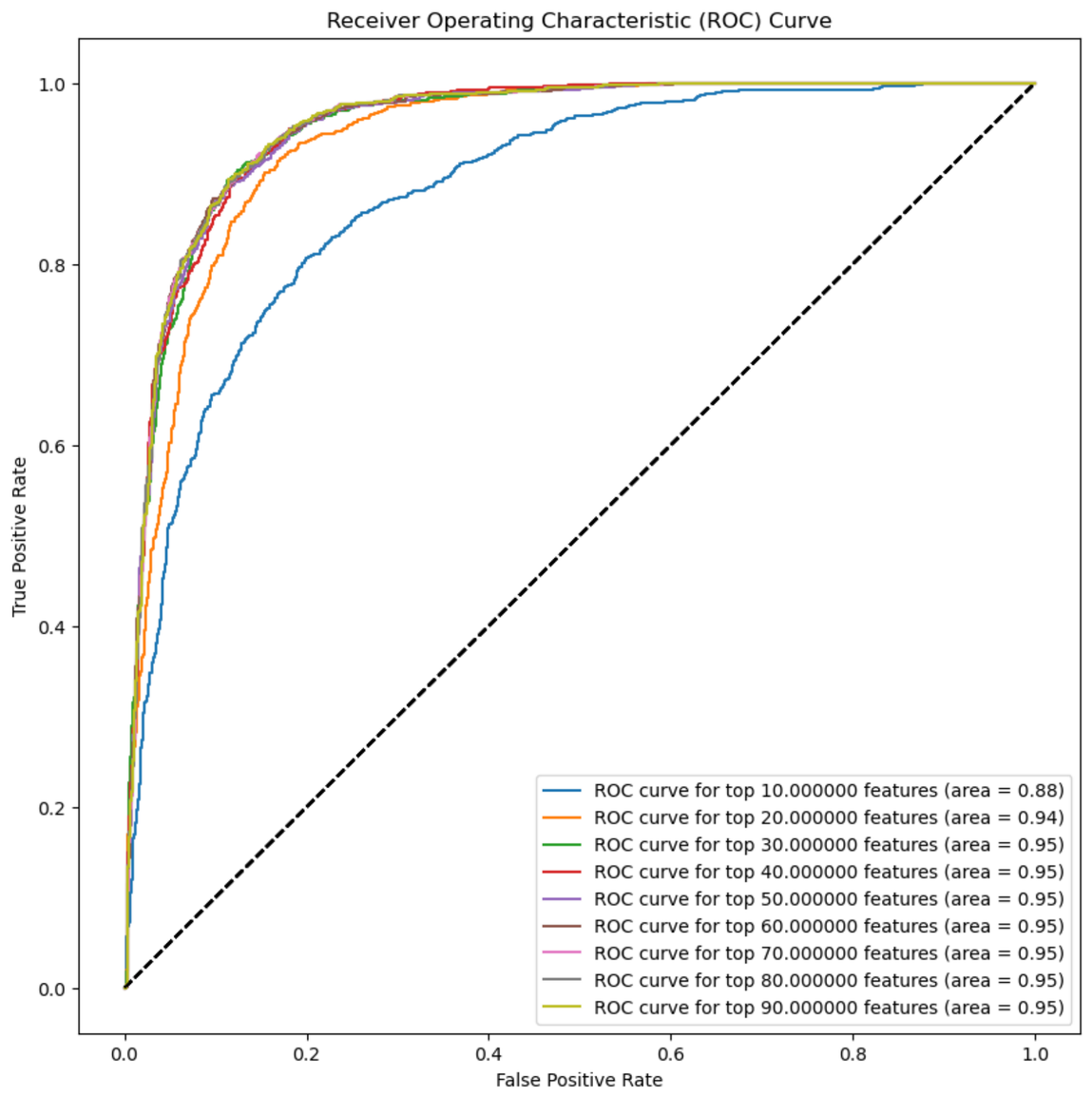
- solver:
  - svd- good for small & large datasets
  - lsqr - good for small datasets
  - eigen - requires positive numbers, small datasets
- no_components (1 for target with 2 classes
- tolereance
- Shrinkage (for small data sets where cols out number rows)
- Store covariance (True or False)

```
X_test_scaled=scaler.transform(X_test_n)

#Model
lda=LinearDiscriminantAnalysis()
# as only 2 outcomes for target, the transforamtion returns one single columns
# Define the hyperparameter grid to search over
param_grid = {'solver': ['svd', 'lsqr'],
              #'shrinkage': ['None'],
              'n_components': [1],
              'store_covariance': [True, False],
              'tol': [1e-4, 1e-5, 1e-6]}

# Perform the grid search
scoring = {'accuracy': 'accuracy',
   'precision_macro': 'precision_macro',
   'recall_macro': 'recall_macro'}
grid_search = GridSearchCV(lda, param_grid=param_grid, cv=5, scoring=scoring, refit=False, verbose=2)
grid_search.fit(X_train_scaled, y_train)
# print('Best hyperparameters:', grid_search.best_params_)
# print('Best cross-validation score:', grid_search.best_score_)

print(grid_search.cv_results_)
results = grid_search.cv_results_
for i in range(len(results['params'])):
    print(f"Model {i+1}: n_components={results['params'][i]['n_components']}, \
        Accuracy={results['mean_test_accuracy'][i]:.3f}, \
```

# ERROR METRICS & ROC




| Error_metric | Test_10 | Test_20 | Test_30 | Test_40 | Test_50 | Test_60 | Test_70 | Test_80 | Test_90 |
|---|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.797727 | 0.867424 | 0.889394 | 0.881818 | 0.881439 | 0.885227 | 0.885227 | 0.884091 | 0.885606 |
| Precision | 0.767313 | 0.864329 | 0.872794 | 0.862774 | 0.862144 | 0.864727 | 0.864198 | 0.863372 | 0.864826 |
| Recall | 0.848392 | 0.868300 | 0.908882 | 0.905054 | 0.905054 | 0.910413 | 0.911179 | 0.909648 | 0.911179 |



# MODEL COMPARISON

## AdaBase Classifer

AdaBase using 80 features

| | Error_metric | Train | Test |
|---|---|---|---|
| 0 | Accuracy | 0.979731 | 0.965152 |
| 1 | Precision | 0.970017 | 0.954341 |
| 2 | Recall | 0.990176 | 0.976263 |

## Bernoulli NB

With 50 features

| | Error_metric | Train | Test |
|---|---|---|---|
| **0** | Accuracy | 0.86 | 0.85 |
| **1** | Precision | 0.82 | 0.81 |
| **2** | Recall | 0.91 | 0.91 |

## Linear Discriminant Analysis

| | Error_metric | Test_30 |
|---|---|---|
| 0 | Accuracy | 0.889394 |
| 1 | Precision | 0.872794 |
| 2 | Recall | 0.908882 |


## Histgradientboost Classifier

| | Error_metric | Train | Test |
|---|---|---|---|
| **0** | Accuracy | 0.998 | 0.988 |
| **1** | Precision | 0.997 | 0.975 |
| **2** | Recall | 1.000 | 1.000 |

# THE END