AI4E - FINAL ASSESSMENT

Student Learning Analysts Project

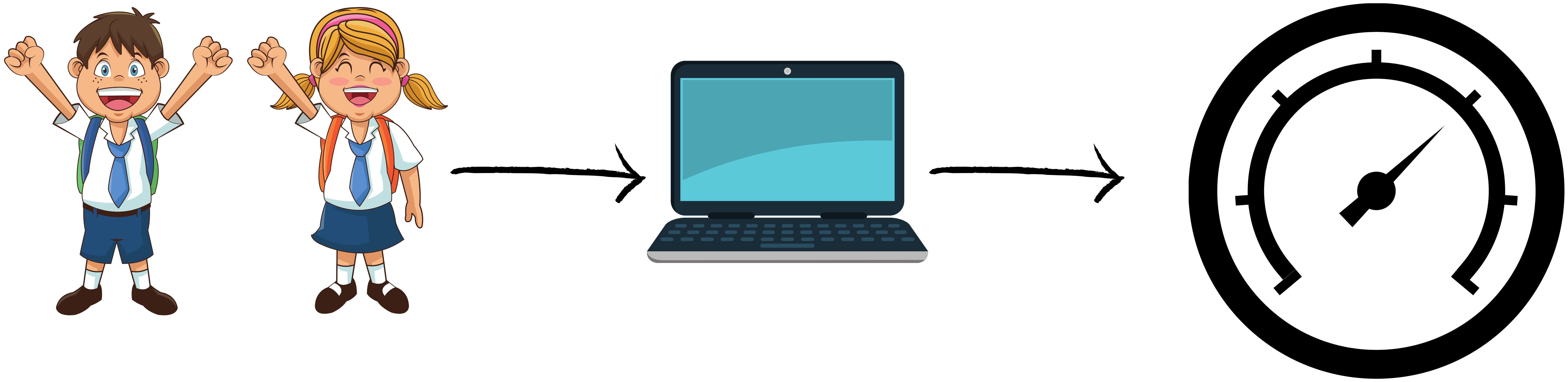
Project Owner : Vu Kim Duy
Lecturer : Nguyễn Thanh Tuấn
Teaching Assisstant : Nguyễn Thành Trung

Appendix

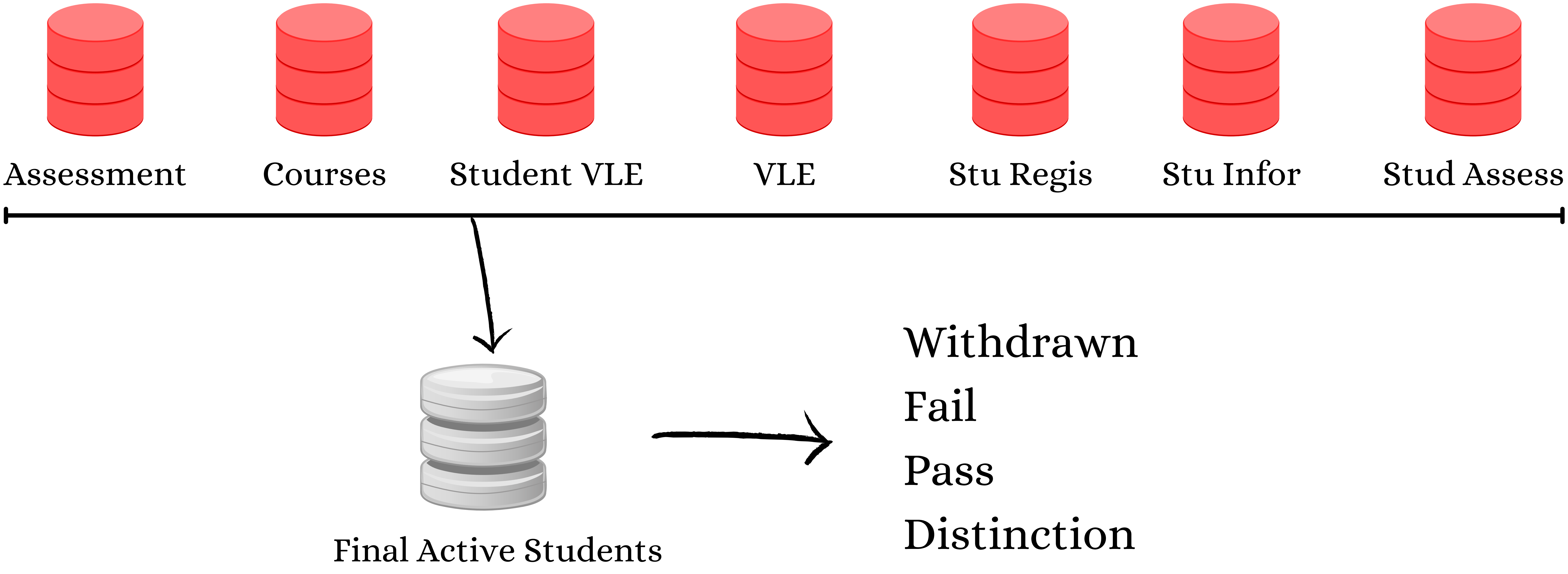
Introduction

Ideas

- Inspired by the 2 days event hold by Learning Analytics & Open Data Hackathon 3.0 Competition at the University of British Columbia, Canada
- Develop the Machine Learning Model to evaluate student's learning performance when interactive with Virtual Learning Environment



Input & Output



Classification Problem

Dataset Exploratory

Student Information

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31284 entries, 0 to 32592
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   code_module           31284 non-null  object
 1   code_presentation     31284 non-null  object
 2   id_student            31284 non-null  int64
 3   gender                31284 non-null  object
 4   region                31284 non-null  object
 5   highest_education     31284 non-null  object
 6   imd_band              31284 non-null  object
 7   age_band              31284 non-null  object
 8   num_of_prev_attempts  31284 non-null  int64
 9   studied_credits       31284 non-null  int64
 10  disability            31284 non-null  object
 11  final_result          31284 non-null  object
dtypes: int64(3), object(9)
memory usage: 4.4+ MB
```

	code_module	code_presentation	id_student	gender	region	highest_education	imd_band	age_band	num_of_prev_attempts	studied_credits	disability	final_result
0	AAA	2013J	11391	M	East Anglian Region	HE Qualification	90-100%	55<=	0	240	N	Pass
1	AAA	2013J	28400	F	Scotland	HE Qualification	20-30%	35-55	0	60	N	Pass
2	AAA	2013J	30268	F	North Western Region	A Level or Equivalent	30-40%	35-55	0	60	Y	Withdrawn
3	AAA	2013J	31604	F	South East Region	A Level or Equivalent	50-60%	35-55	0	60	N	Pass
4	AAA	2013J	32885	F	West Midlands Region	Lower Than A Level	50-60%	0-35	0	60	N	Pass

Student VLE

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10655280 entries, 0 to 10655279
Data columns (total 6 columns):
 #   Column             Dtype 
---  ------             ----- 
 0   code_module        object
 1   code_presentation  object
 2   id_student         int64 
 3   id_site            int64 
 4   date               int64 
 5   sum_click          int64 
dtypes: int64(4), object(2)
memory usage: 487.8+ MB
```

	code_module	code_presentation	id_student	id_site	date	sum_click
0	AAA	2013J	28400	546652	-10	4
1	AAA	2013J	28400	546652	-10	1
2	AAA	2013J	28400	546652	-10	1
3	AAA	2013J	28400	546614	-10	11
4	AAA	2013J	28400	546714	-10	1

Assessment

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 206 entries, 0 to 205
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   code_module         206 non-null    object
 1   code_presentation   206 non-null    object
 2   id_assessment       206 non-null    int64
 3   assessment_type     206 non-null    object
 4   date                195 non-null    float64
 5   weight              206 non-null    float64
dtypes: float64(2), int64(1), object(3)
memory usage: 9.8+ KB
```

	code_module	code_presentation	id_assessment	assessment_type	date	weight
0	AAA	2013J	1752	TMA	19.0	10.0
1	AAA	2013J	1753	TMA	54.0	20.0
2	AAA	2013J	1754	TMA	117.0	20.0
3	AAA	2013J	1755	TMA	166.0	20.0
4	AAA	2013J	1756	TMA	215.0	30.0

Student Assessments

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 173912 entries, 0 to 173911
Data columns (total 5 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   id_assessment   173912 non-null  int64
 1   id_student      173912 non-null  int64
 2   date_submitted  173912 non-null  int64
 3   is_banked       173912 non-null  int64
 4   score           173739 non-null  float64
dtypes: float64(1), int64(4)
memory usage: 6.6 MB
```

	id_assessment	id_student	date_submitted	is_banked	score
0	1752	11391	18	0	78.0
1	1752	28400	22	0	70.0
2	1752	31604	17	0	72.0
3	1752	32885	26	0	69.0
4	1752	38053	19	0	79.0

After exploring features from other tables

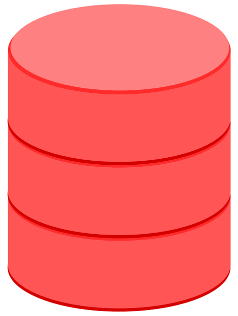
- Firstly, I choose Student Information & Student VLE to explore since they are correspondent with number of interactions of individdual student

Stu Infor → Actual Student : 28785

Stu VLE → Interacts with VLE: 26074 (Has "click" records)

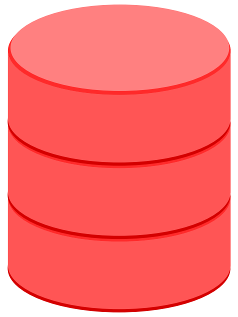
→ **df_merge_stuVle_stuIn** (contains both student information and total clicks of 1 student to records

- Secondly, I choose Student Assessments & Assessments to explore since they are mutating with the record of each assessment



Student Assessments

Containts potential features : score, id_assessment, id_student

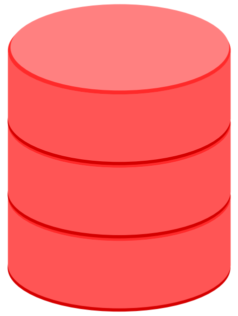


Assessments

Contains all assessment in 1 module (has keys: id_assessment, code_module, code_presentation)

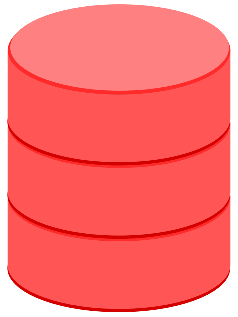
df_stuAss_record (contains the entire assessments of 1 student delivered in 1 module)

- Finally, I joined 2 important data frame by far to release the final data frame for training model and further analysis



Contains all records of student's assessments (keys to join: id_student, code_module)

df_stuAss_record



Contains all records of student's background & interaction with VLE (keys to join: code_module, id_student)

df_merge_stuVle_stuIn

df_final_active_stu

Final Active Student Dataframe

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31284 entries, 0 to 31283
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   code_module           31284 non-null  object
 1   code_presentation     31284 non-null  object
 2   id_student            31284 non-null  int64
 3   gender                31284 non-null  object
 4   region                31284 non-null  object
 5   highest_education     31284 non-null  object
 6   imd_band              31284 non-null  object
 7   age_band              31284 non-null  object
 8   num_of_prev_attempts  31284 non-null  int64
 9   studied_credits       31284 non-null  int64
 10  disability            31284 non-null  object
 11  final_result          31284 non-null  object
 12  sum_click             31284 non-null  float64
 13  mean_score            25067 non-null  float64
dtypes: float64(2), int64(3), object(9)
memory usage: 3.6+ MB
```

	code_module	code_presentation	id_student	gender	region	highest_education	imd_band	age_band	num_of_prev_attempts	studied_credits	disability	final_result	sum_click	mean_score
0	AAA	2013J	11391	M	East Anglian Region	HE Qualification	90-100%	55<=	0	240	N	Pass	934.0	82.0
1	AAA	2013J	28400	F	Scotland	HE Qualification	20-30%	35-55	0	60	N	Pass	1435.0	66.4
2	AAA	2013J	30268	F	North Western Region	A Level or Equivalent	30-40%	35-55	0	60	Y	Withdrawn	281.0	NaN
3	AAA	2013J	31604	F	South East Region	A Level or Equivalent	50-60%	35-55	0	60	N	Pass	2158.0	76.0
4	AAA	2013J	32885	F	West Midlands Region	Lower Than A Level	50-60%	0-35	0	60	N	Pass	1034.0	54.4

Data Analysis Exploratory

Remove Redundant Columns

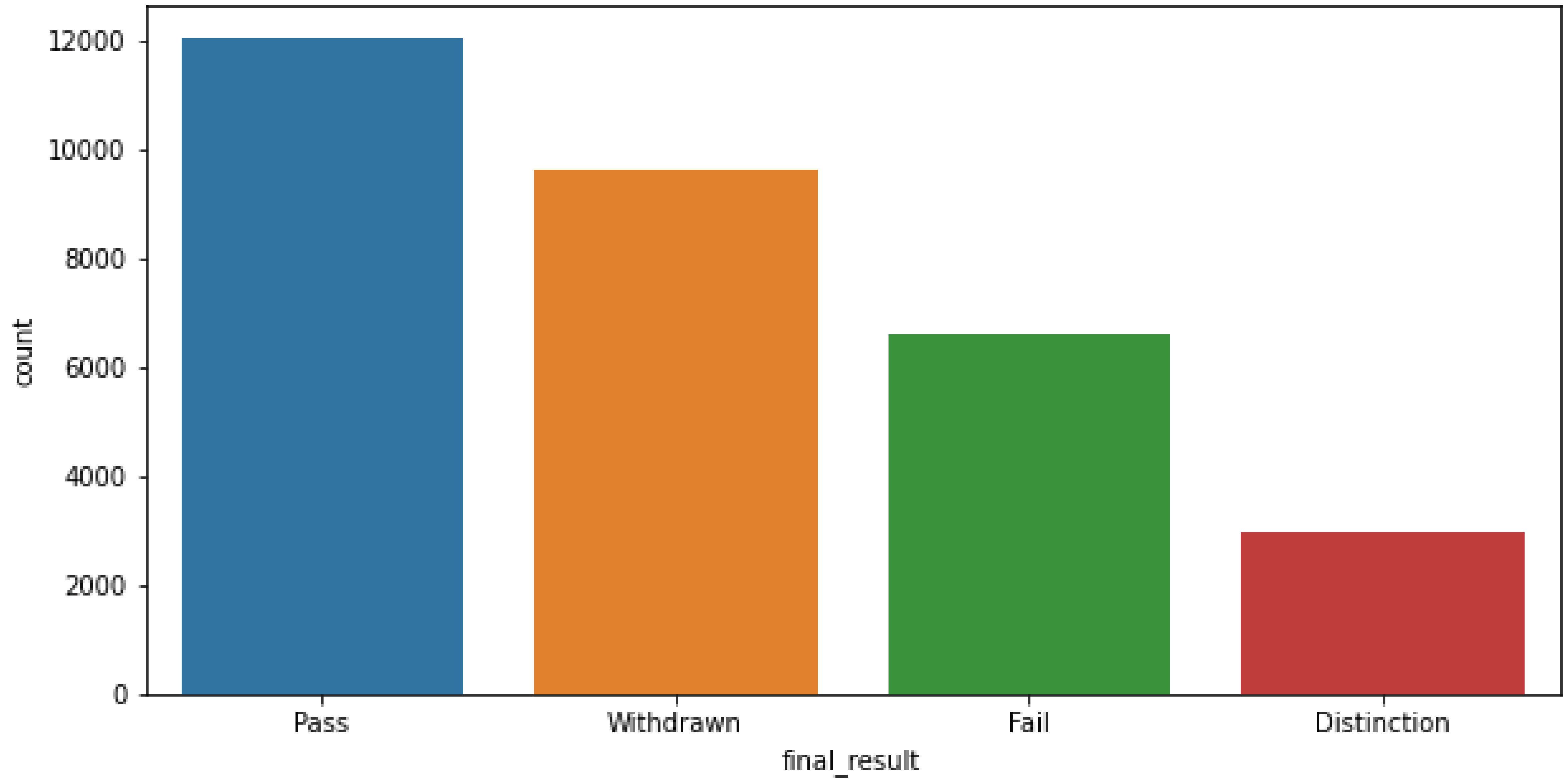
- **code_module**: contains a representative notation of 1 module. Eg: AAA, BBB
- **code_presentation**: contains a representative notation of 1 presentation. Eg: 2013J
- **id_student**: anonymized data
- **num_of_prev_attempts**: number of re-attempt to sit an assessment
- **studied_credits**: total credits student achieved in 1 module

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31284 entries, 0 to 31283
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             31284 non-null  object
 1   region             31284 non-null  object
 2   highest_education  31284 non-null  object
 3   imd_band           31284 non-null  object
 4   age_band           31284 non-null  object
 5   disability         31284 non-null  object
 6   final_result       31284 non-null  object
 7   sum_click          31284 non-null  float64
 8   mean_score         25067 non-null  float64
dtypes: float64(2), object(7)
memory usage: 2.4+ MB
```

	gender	region	highest_education	imd_band	age_band	disability	final_result	sum_click	mean_score
0	M	East Anglian Region	HE Qualification	90-100%	55<=	N	Pass	934.0	82.0
1	F	Scotland	HE Qualification	20-30%	35-55	N	Pass	1435.0	66.4
2	F	North Western Region	A Level or Equivalent	30-40%	35-55	Y	Withdrawn	281.0	NaN
3	F	South East Region	A Level or Equivalent	50-60%	35-55	N	Pass	2158.0	76.0
4	F	West Midlands Region	Lower Than A Level	50-60%	0-35	N	Pass	1034.0	54.4

Target Column

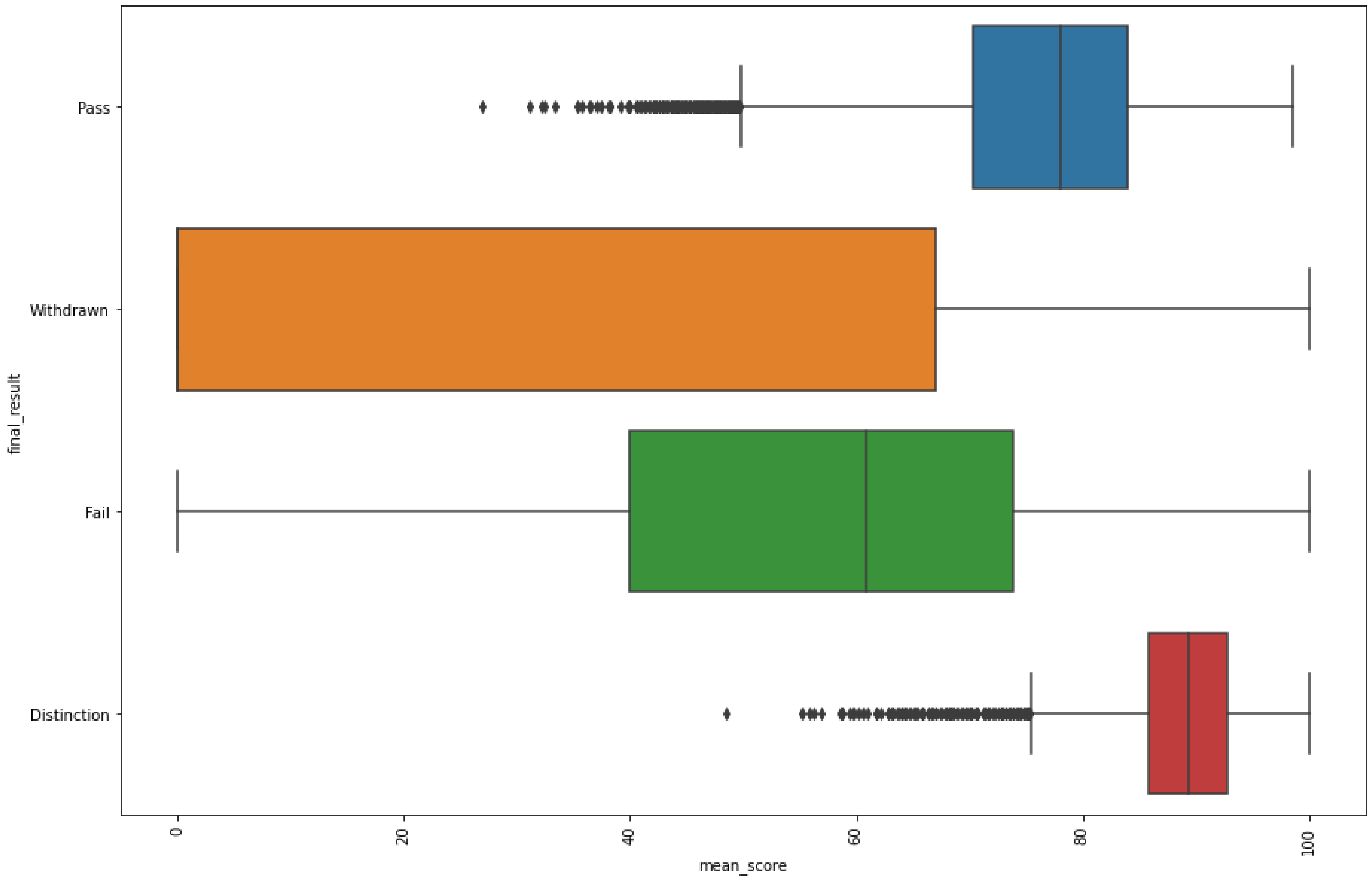
- Notice that our target column is imbalanced on 4 difference classes
- Could affect to our model's performance since the process of splitting dataset will deliver unequal quantity on each class



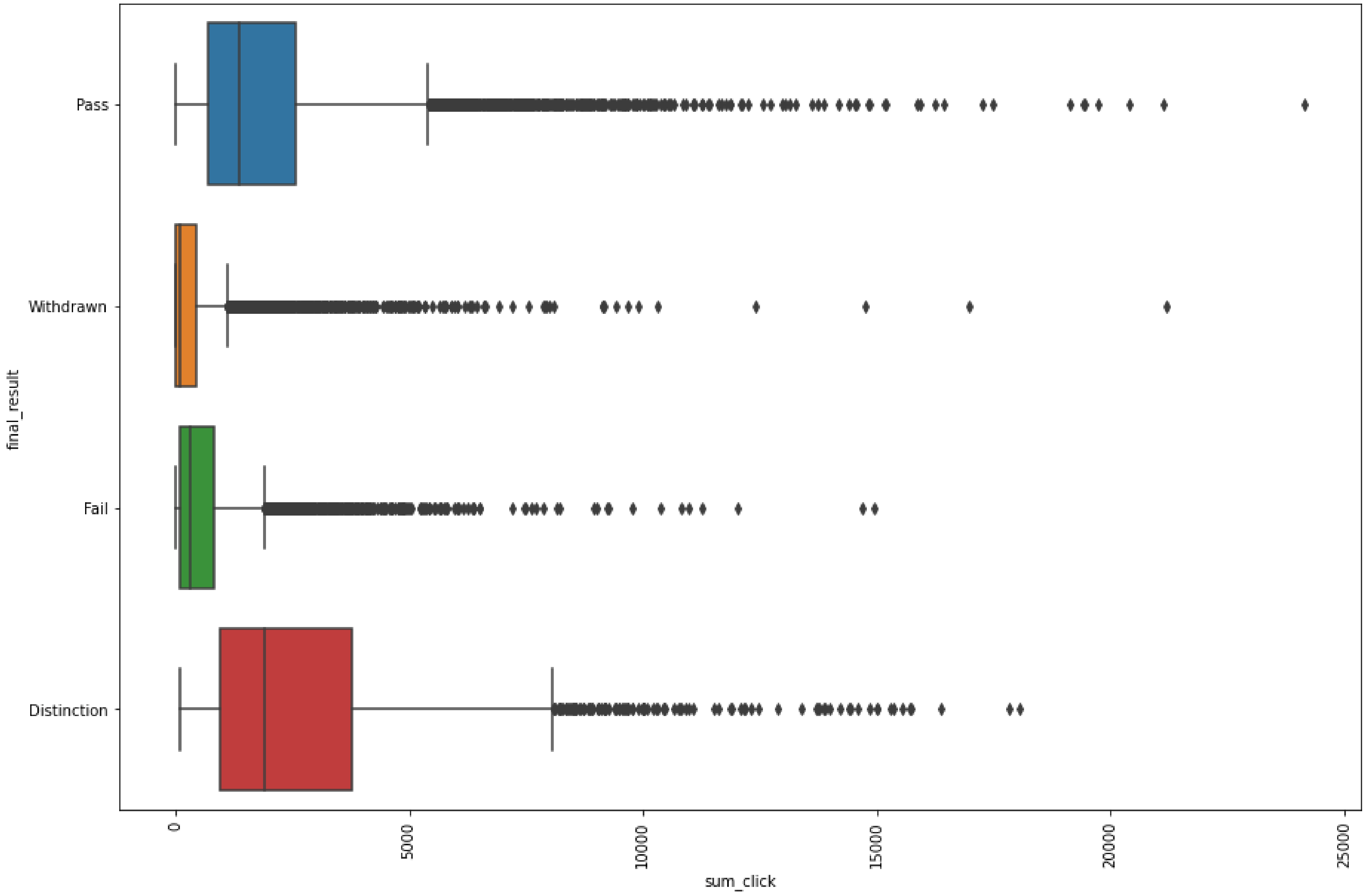
Potential Features and Final Result

- **mean_score**: Mean score of 1 student who participates in all assessments (contains NaN values)
    - **Withdrawn**: filter with 0
    - **Fail**: filter with 39
    - **Pass**: filter with 40
    - **Distinction**: No missing values

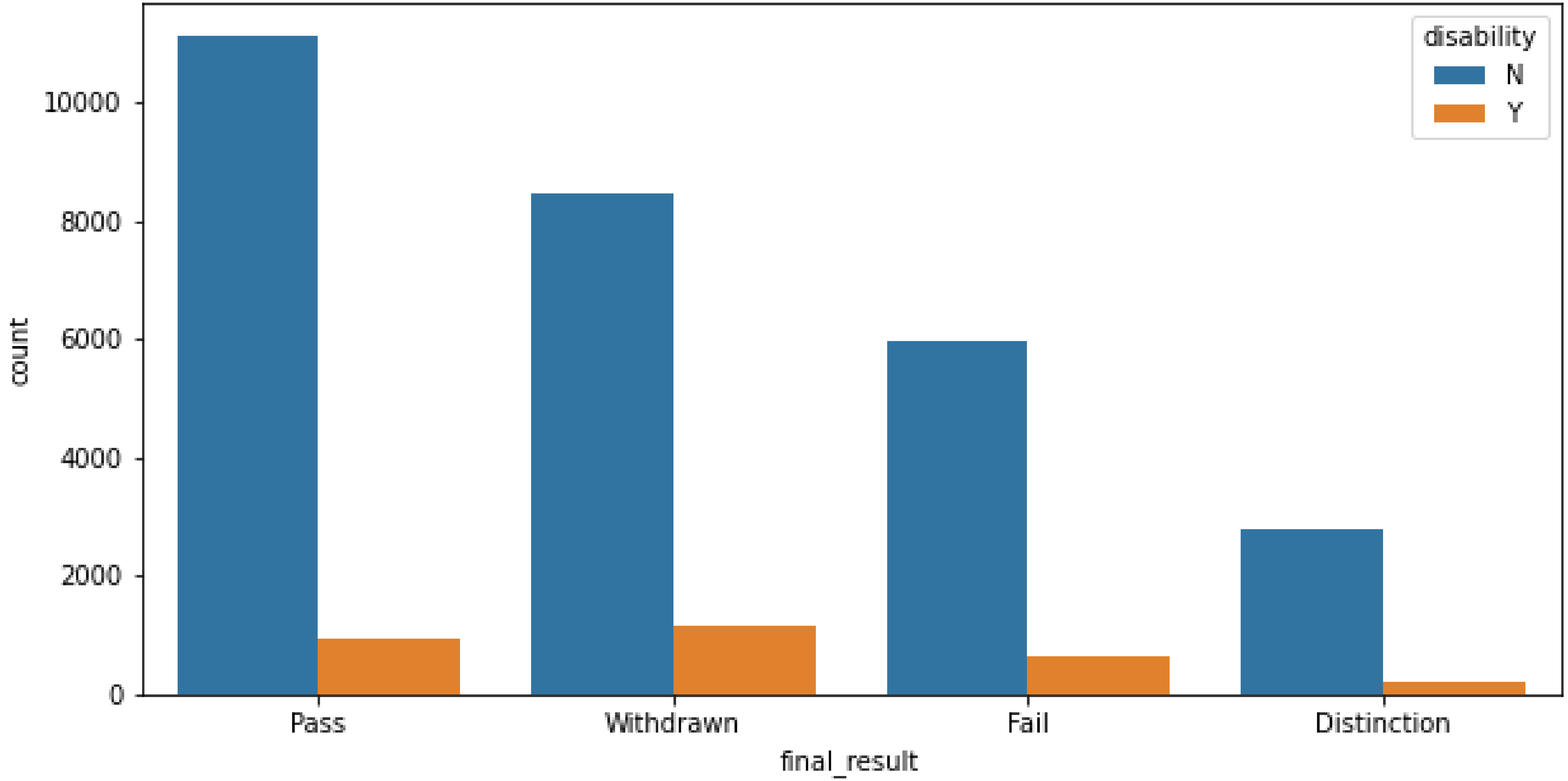
```
Outliers of Distinction: 4.66%
Outliers of Pass: 1.83%
Outliers of Withdrawn: 0.00%
Outliers of Fail: 0.00%
```



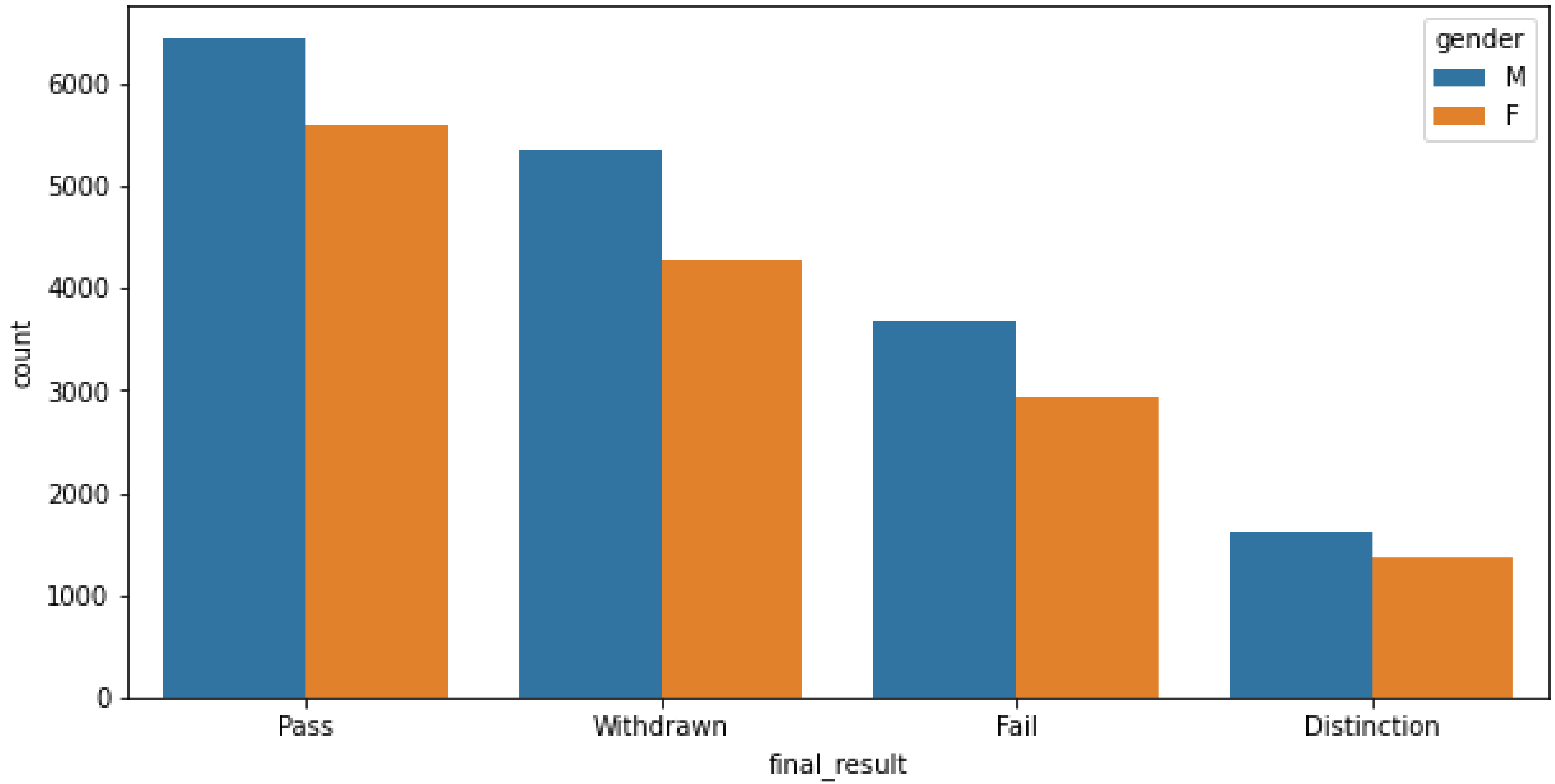
- **sum_click**: Contains a total of clicks of 1 student in 1 module (contains NaN values)
  - **NaN values:** practically, happens when FAIL or WITHDRAWN student does not interact with VLE
  - **Fill with 0**



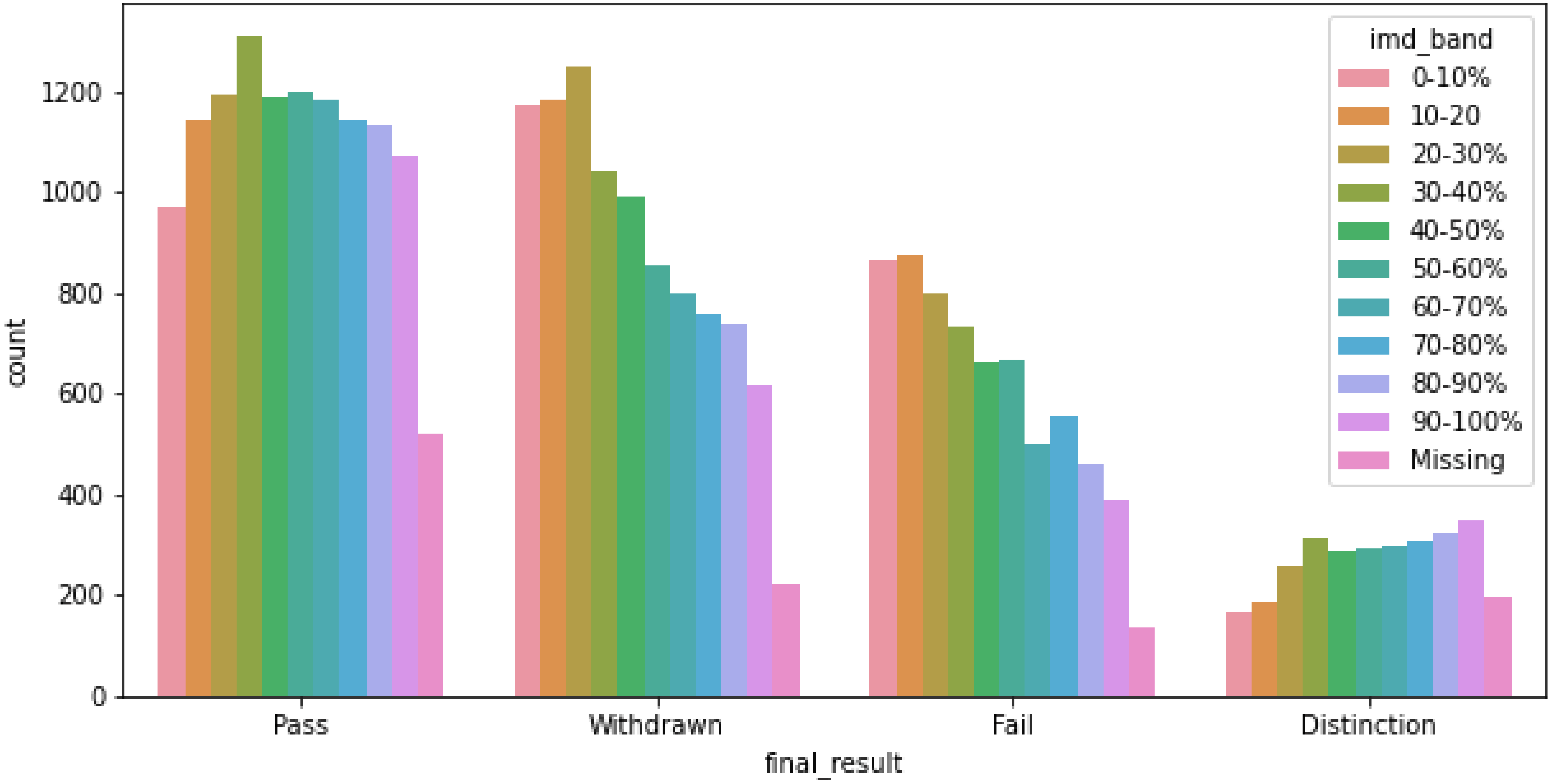
- **disability**: Determines weather it's a disable student or normal student
    - No special relationship to final_result



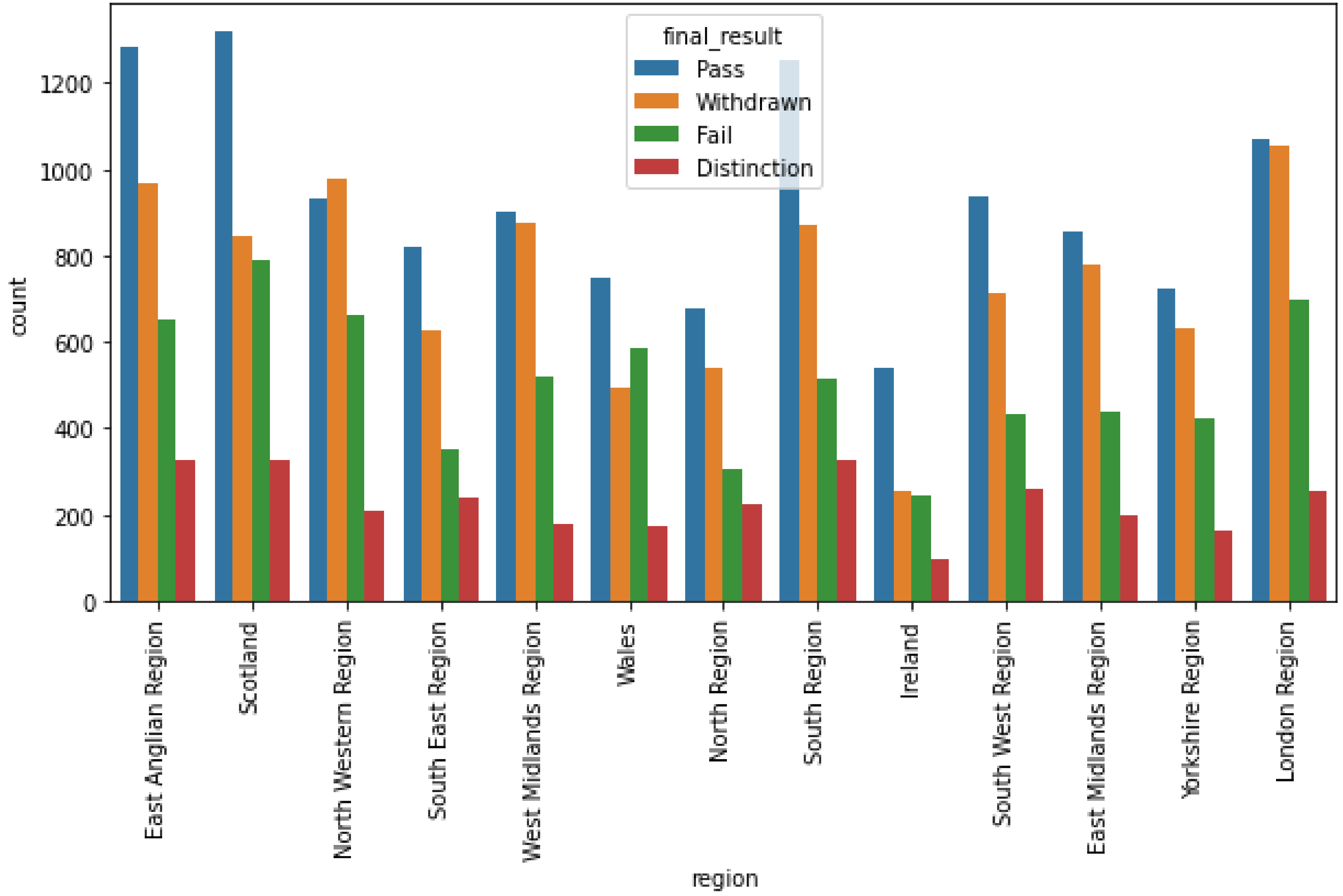
- **gender**: Determines the sex of a student



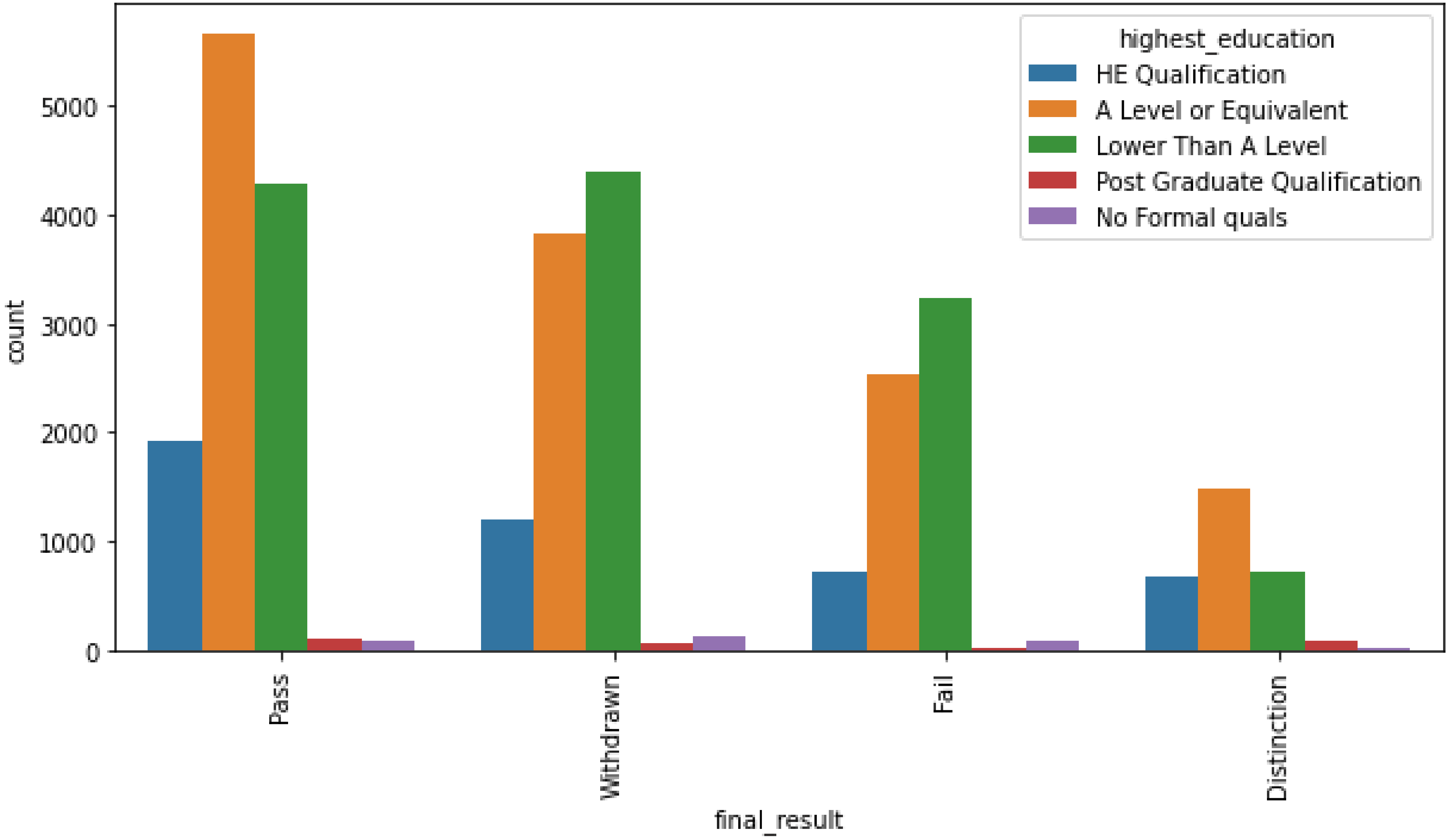
- **imd_band:** Shorts for Indices of Multiple Deprivation
    - Determines how derivative is of some areas in England measured by the range of percentages



- **Region:** Determines the location of sitting an exam of 1 student in England



- **Highest Education:** Determines the current education of 1 student when enrolling the module



Building Models On Imbalanced Classes

Encoding Models

- Since our data frame contains mostly categorical values
- Apply 2 techniques of encoding: **Label Encoder** & **One-hot Encoder**
- **Label Encoder**
  - Columns to be applied: final_result
- **One-hot Encoder**
  - Using **get_dummies()** function
  - Columns to be applied: region, highest_education, imd_band, age_band, disability, gender

get_dummies() : For One - Hot Encoding

	sum_click	mean_score	region_East Midlands Region	region_Ireland	region_London Region	region_North Region	region_North Western Region	region_Scotland	region_South East Region	region_South Region	region_South West Region	region_Wales	region_ Midl Re
0	934.0	82.0	0	0	0	0	0	0	0	0	0	0	
1	1435.0	66.4	0	0	0	0	0	1	0	0	0	0	
2	281.0	0.0	0	0	0	0	1	0	0	0	0	0	
3	2158.0	76.0	0	0	0	0	0	0	1	0	0	0	
4	1034.0	54.4	0	0	0	0	0	0	0	0	0	0	

LabelEncoder() : For ordinal values

```
array([2, 2, 3, ..., 2, 3, 0])
```

Train / Validation Split

- Split our data frame into 2 parts which contain random sample for testing and training
- Avoid data leakage and perform equally on the training model

```
validation_size = 0.2
seed   = 42
X_train, X_validation, y_train, y_validation = train_test_split(X, y, test_size=validation_size, random_state=seed)
```

Apply Various Models (on Imbalanced Classes)

- I will apply the training dataset to 5 main models for the multi-classification problems
  - Logistic Regression
  - K - Neighbors Classifier
  - Decision Tree Classifier
  - Gaussian Naive Bayes
  - Support Vector Machine
- The reason for training on different models is to decide the best performing model
- Split into 2 scenarios : **Scaled Features** and **Non - scaled Features**

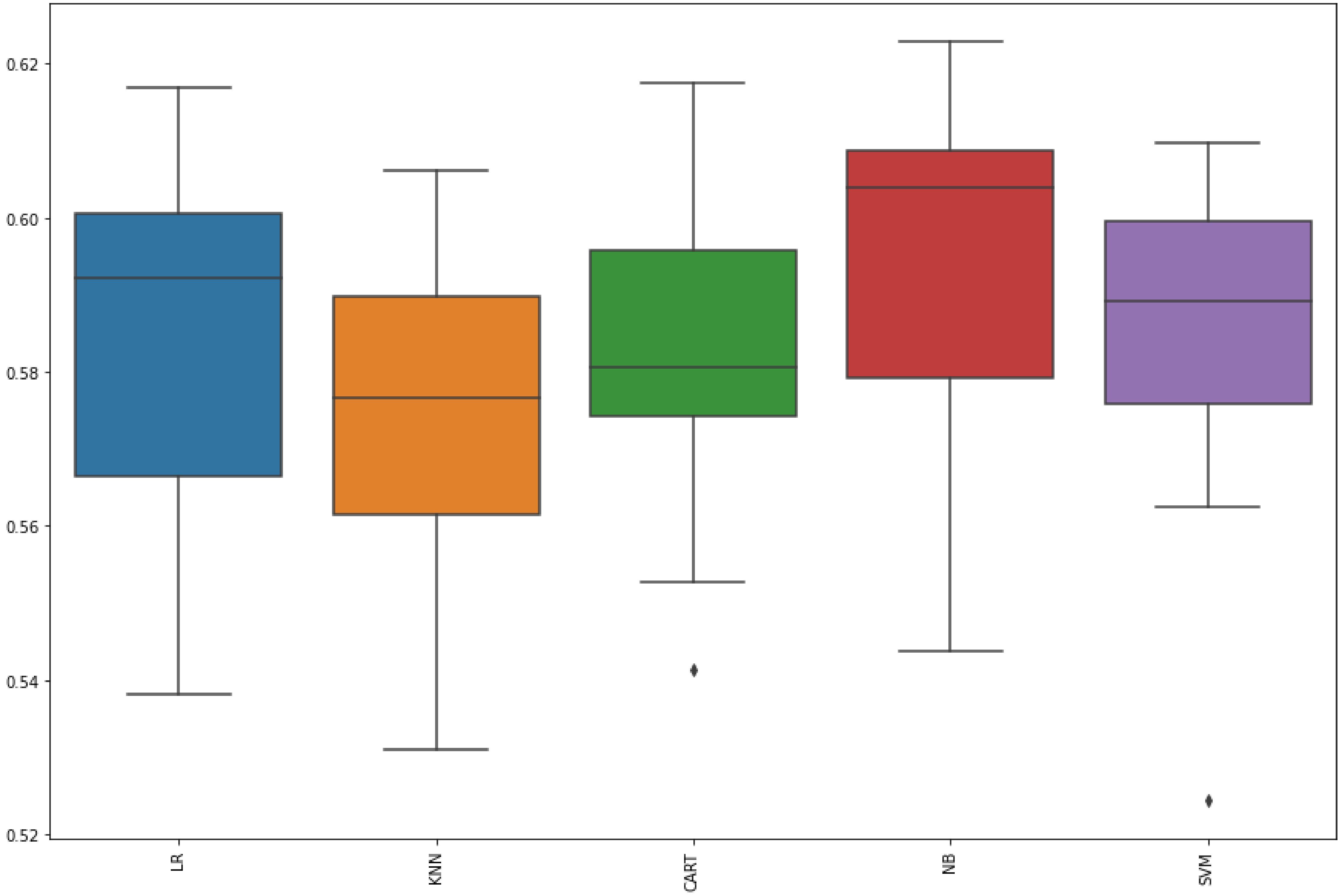
- **Model on Non - Scaled Features**
  - Using K - Fold Cross Validation to avoid overfitting

```
kfold = KFold(n_splits=num_folds, random_state=seed, shuffle=True)
cv_results = cross_val_score(model,X_train, y_train, cv=kfold, scoring=scoring)
results.append(cv_results)
names.append(name)
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)
```

  - **Models Comparision**
    - KNN performs better than other but the accuracy is quite low

```
LR: 0.594998 (0.009818)
KNN: 0.610940 (0.006576)
CART: 0.575060 (0.009205)
NB: 0.543693 (0.010655)
SVM: 0.590003 (0.011398)
```

- Presenting Model Performance on Boxplot for better visualization



- **Models On Scaled Features**
  - **Using Standard Scaler**
  - **mean_score, sum_click** has distinguished values, which is the reason why I implement **feature scaling**

	sum_click	mean_score
0	934.0	82.0
1	1435.0	66.4
2	281.0	0.0
3	2158.0	76.0
4	1034.0	54.4

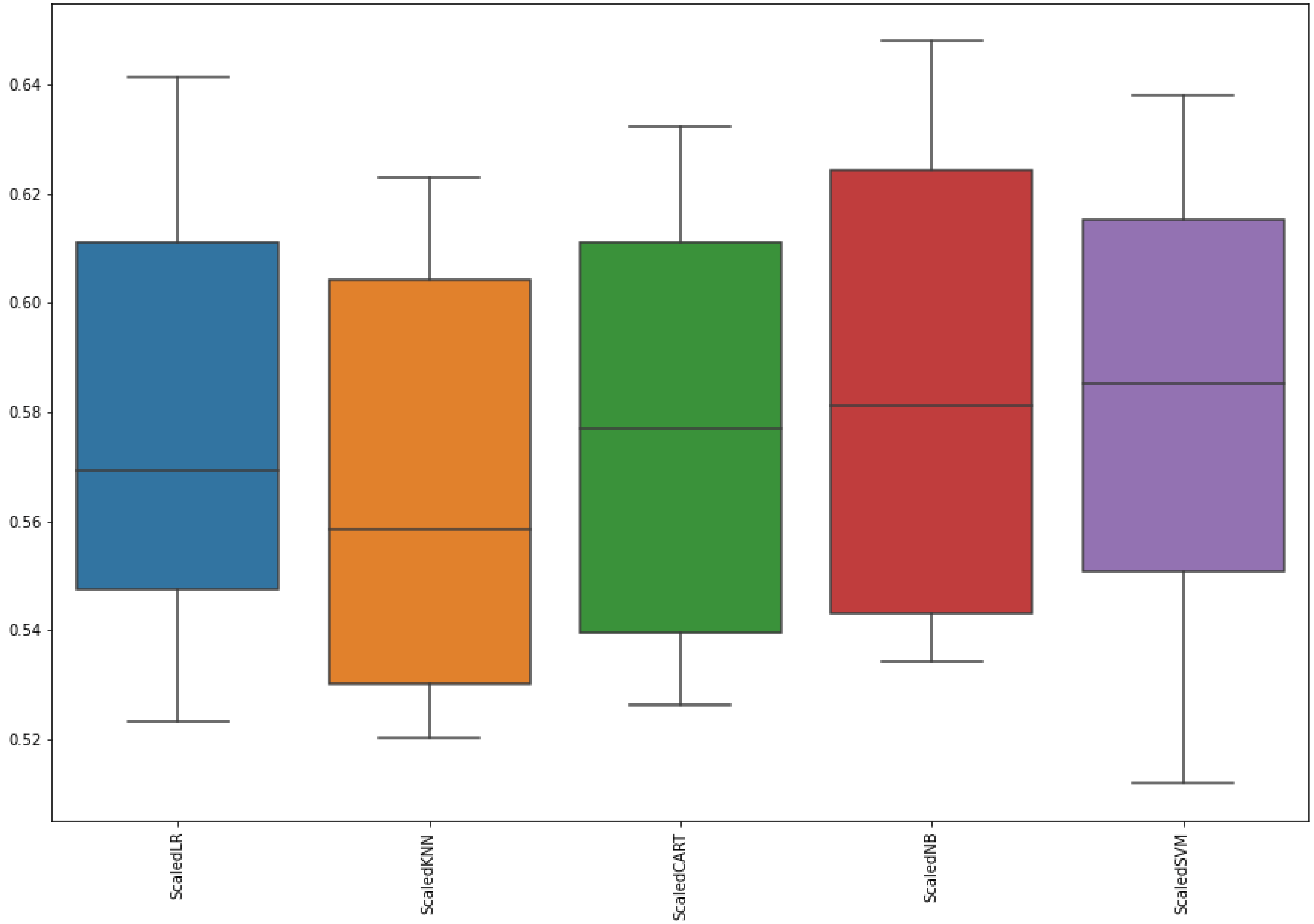
  - Using **K - Fold Cross Validation** to avoid overfitting

```
for name, model in models:
  kfold = KFold(n_splits=num_folds, random_state=seed, shuffle=True)
  cv_results = cross_val_score(model,X_train, y_train, cv=kfold, scoring=scoring)
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)
```

- **Models Comparision**
  - Logistic Regression and SVM scores well mean accuracy but it's worst (~ 62%)

```
ScaledLR: 0.611180 (0.008778)
ScaledKNN: 0.539537 (0.008247)
ScaledCART: 0.574181 (0.010903)
ScaledNB: 0.533264 (0.012341)
ScaledSVM: 0.629440 (0.011784)
```

- Presenting Model Performance on Boxplot for better visualization



Choosing the final model

- **Logistic Regression**
  - Using GridSearchCV + Pipeline to find the best hyper-parameter for training model
  - Print model's performance on different reports
- **Gaussian Naive Bayes** (On Progress)
  - Due to the low approximate accuracy among models, I decide not to implement GNB at the moment
  - Using GridSearchCV + Pipeline to find the best hyper-parameter for training model
  - Print model's performance on different reports

Logistic Regression

- Setting Pipeline

```
std_slc = StandardScaler()
logistic_Reg = LogisticRegression()


logreg_pipeline = Pipeline(steps=[("std_slc", std_slc),
                                  ("logistic_Reg", logistic_Reg)])
```

- Setting GridSearchCV : find best C and Penalty

```
C = np.logspace(-4, 4, 50)
penalty = ["l1", "l2"]

parameters = dict(logistic_Reg__C=C,
                  logistic_Reg__penalty=penalty)


grid = GridSearchCV(logreg_pipeline, parameters)


grid.fit(X_train, y_train)
```

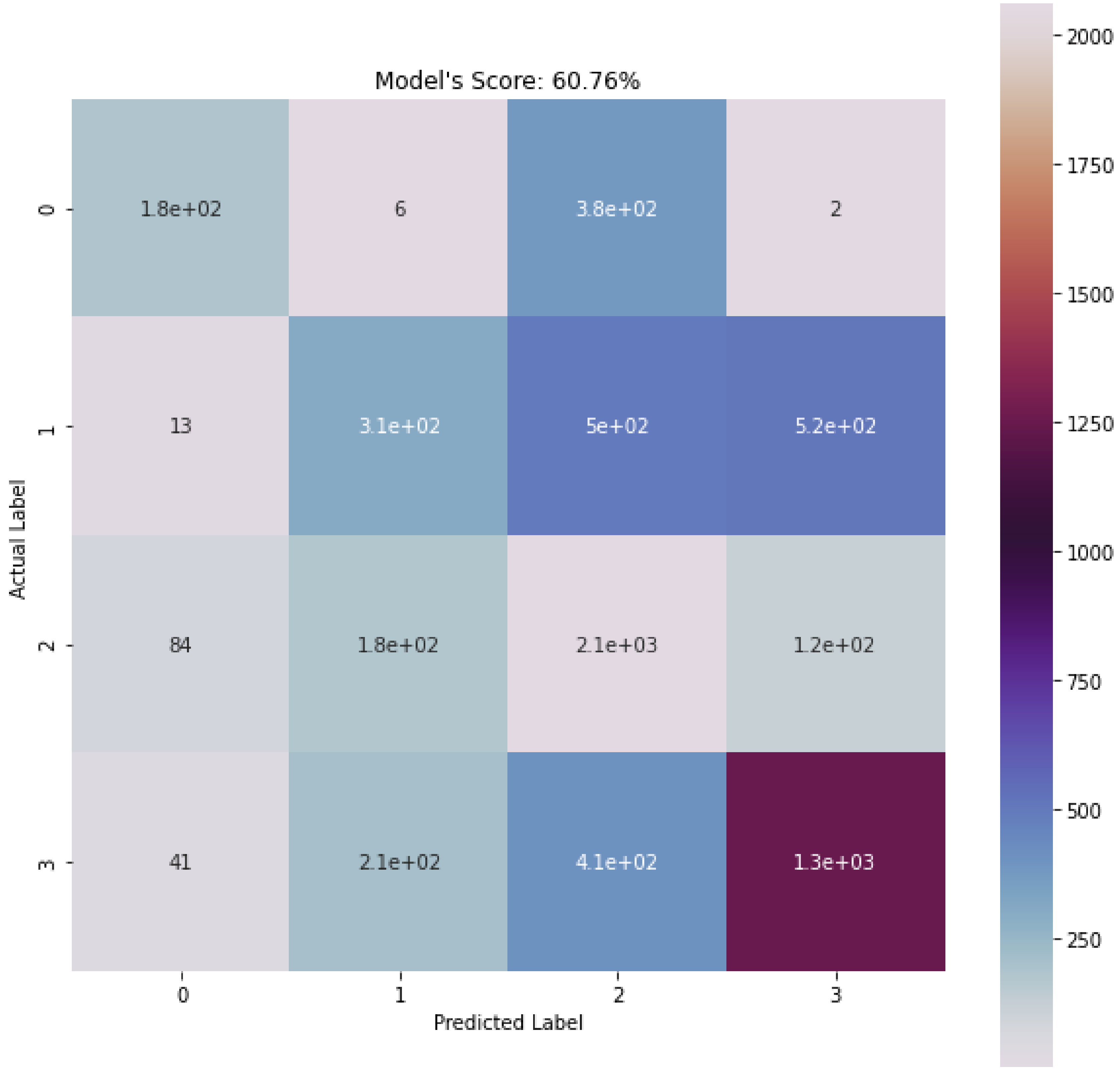
- Best C and Penalty

```
Best Penalty:  l2
Best C:  0.5689866029018293
LogisticRegression(C=0.5689866029018293, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

- Reports

```
Accuracy Score on Validation: 56.27%
Accuracy Score on Training: 60.27%



[[  16   24  521    9]
 [   4  367  715  247]
 [  19  259 2073   92]
 [   5  230  611 1065]]



              precision    recall  f1-score   support

           0       0.36      0.03      0.05       570
           1       0.42      0.28      0.33      1333
           2       0.53      0.85      0.65      2443
           3       0.75      0.56      0.64      1911

    accuracy                           0.56      6257
   macro avg       0.52      0.43      0.42      6257
weighted avg       0.56      0.56      0.53      6257
```

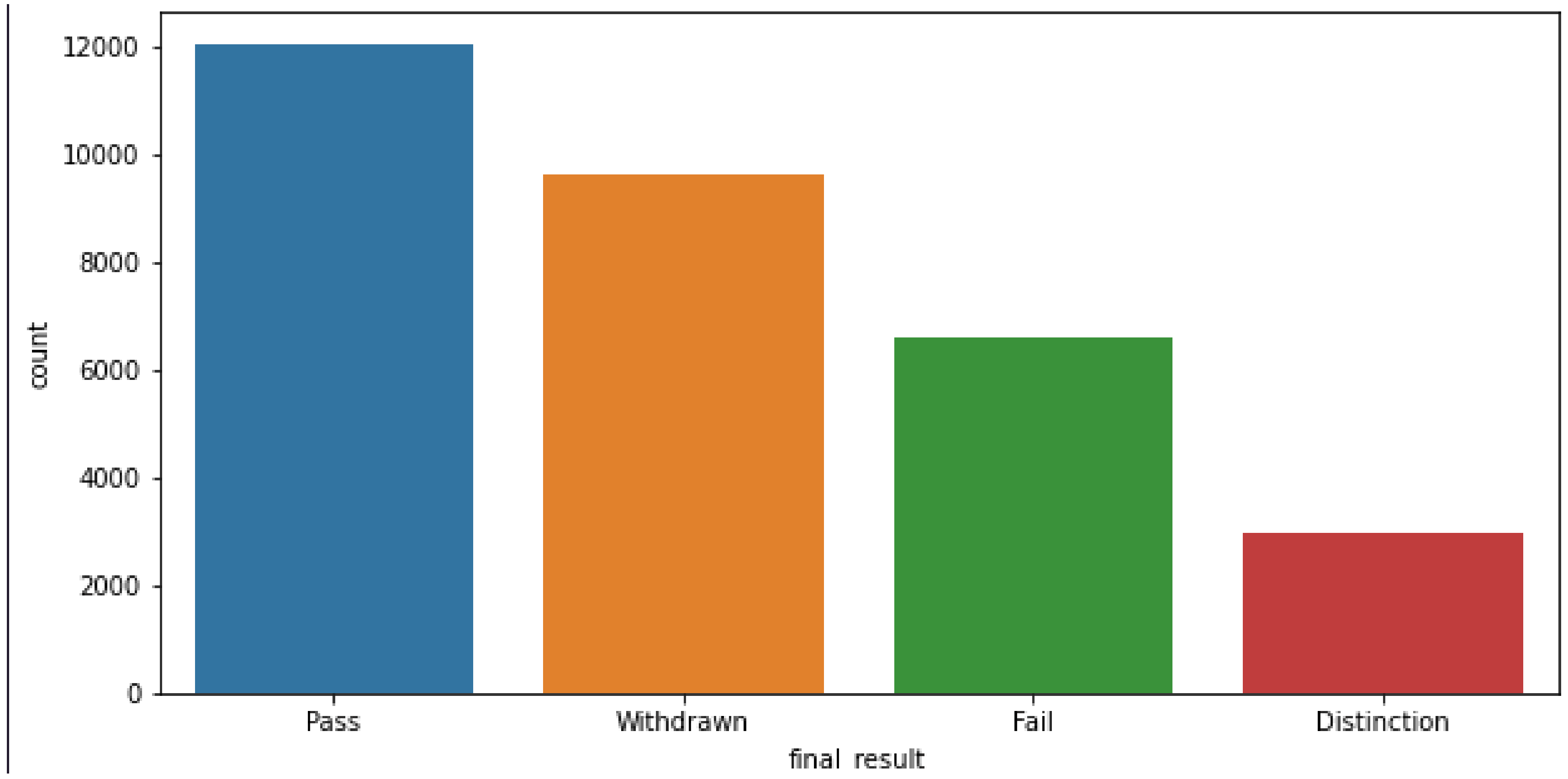
- Represent model's performance on heatmap

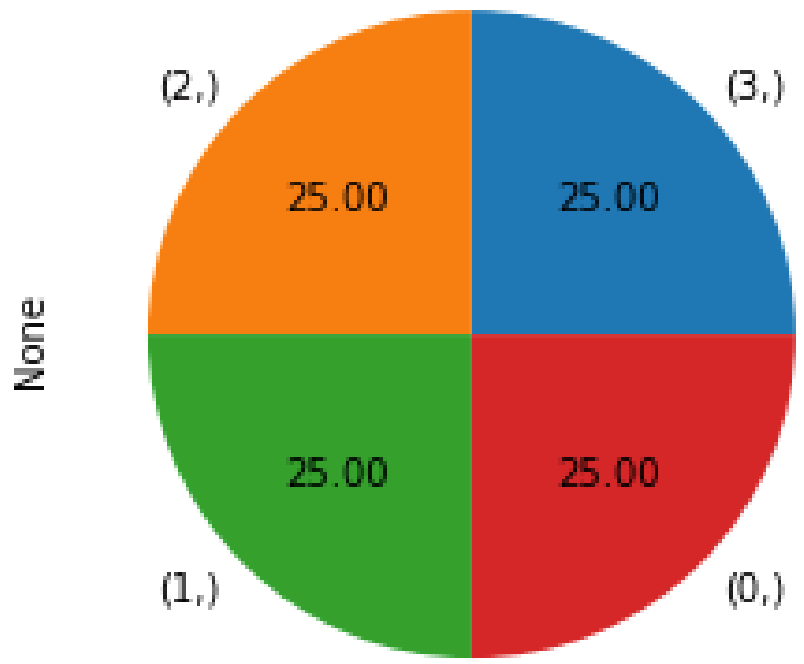


Building Models On Balanced Classes

Apply Various Models (on Balanced Classes)

- Follow the same process as the previous section
- Perform Model Comparison on **Scaled Features** and **Balanced Classes**
- **Resample the Target Column**
    - **Using RandomOverSampler**





- **Scale Feature and Perform Model Comparision**
  - Apply Pipeline
  - Apply K - Fold Cross Validation to avoid overfitting

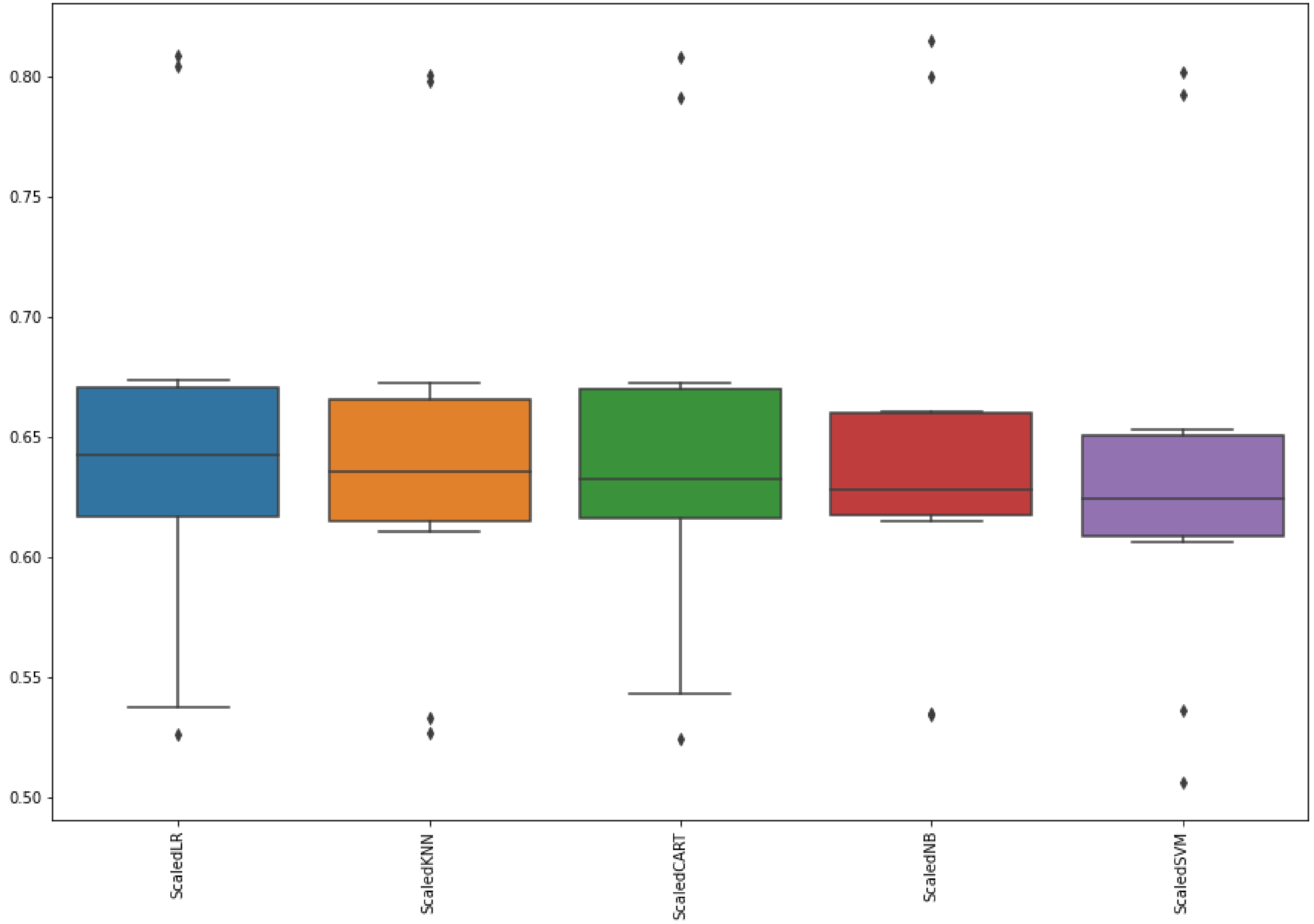
```
pipelines = []
pipelines.append(('ScaledLR', Pipeline([('Scaler', StandardScaler()),('LR',LogisticRegression())])))
pipelines.append(('ScaledKNN', Pipeline([('Scaler', StandardScaler()),('KNN', KNeighborsClassifier())])))
pipelines.append(('ScaledCART', Pipeline([('Scaler', StandardScaler()),('CART', DecisionTreeClassifier())])))
pipelines.append(('ScaledNB', Pipeline([('Scaler', StandardScaler()),('NB',GaussianNB())])))
pipelines.append(('ScaledSVM', Pipeline([('Scaler', StandardScaler()),('SVM', SVC())])))
```

```
results = []
names =  []

for name, model in pipelines:
  kfold = KFold(n_splits=num_folds, random_state=seed, shuffle=True)
  cv_results = cross_val_score(model,resample_X_train, resample_y_train, cv=kfold, scoring=scoring)
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)
```

- **Model Comparision**
    - Decision Tree Classifier performs well on balanced dataset

```
ScaledLR: 0.617867 (0.006677)
ScaledKNN: 0.632028 (0.007965)
ScaledCART: 0.801905 (0.006914)
ScaledNB: 0.530196 (0.009717)
ScaledSVM: 0.660245 (0.010168)
```

- Presenting Models on BoxPlot
  - **Decision Tree Classifier** performs well on balanced dataset



Choosing Final Model

- **Decision Tree Classifier**
  - Setting Pipeline
  - Setting GridSearchCV
  - Print model's performance on different reports

Decision Tree Classifier

- Setting Pipeline

```
std_slc = StandardScaler()
dec = DecisionTreeClassifier()

dec_pipeline = Pipeline(steps=[("std_slc", std_slc),
                               ("dec", dec)])
```

- Setting GridSearchCV

```
param_dict = dict(
    dec__criterion = ["gini", "entropy"],
    dec__max_depth = range(1,10),
    dec__min_samples_split = range(1, 10),
    dec__min_samples_leaf = range(1, 5)
)

grid = GridSearchCV(dec_pipeline,param_dict,cv=10)
grid.fit(resample_X_train, resample_y_train)
```

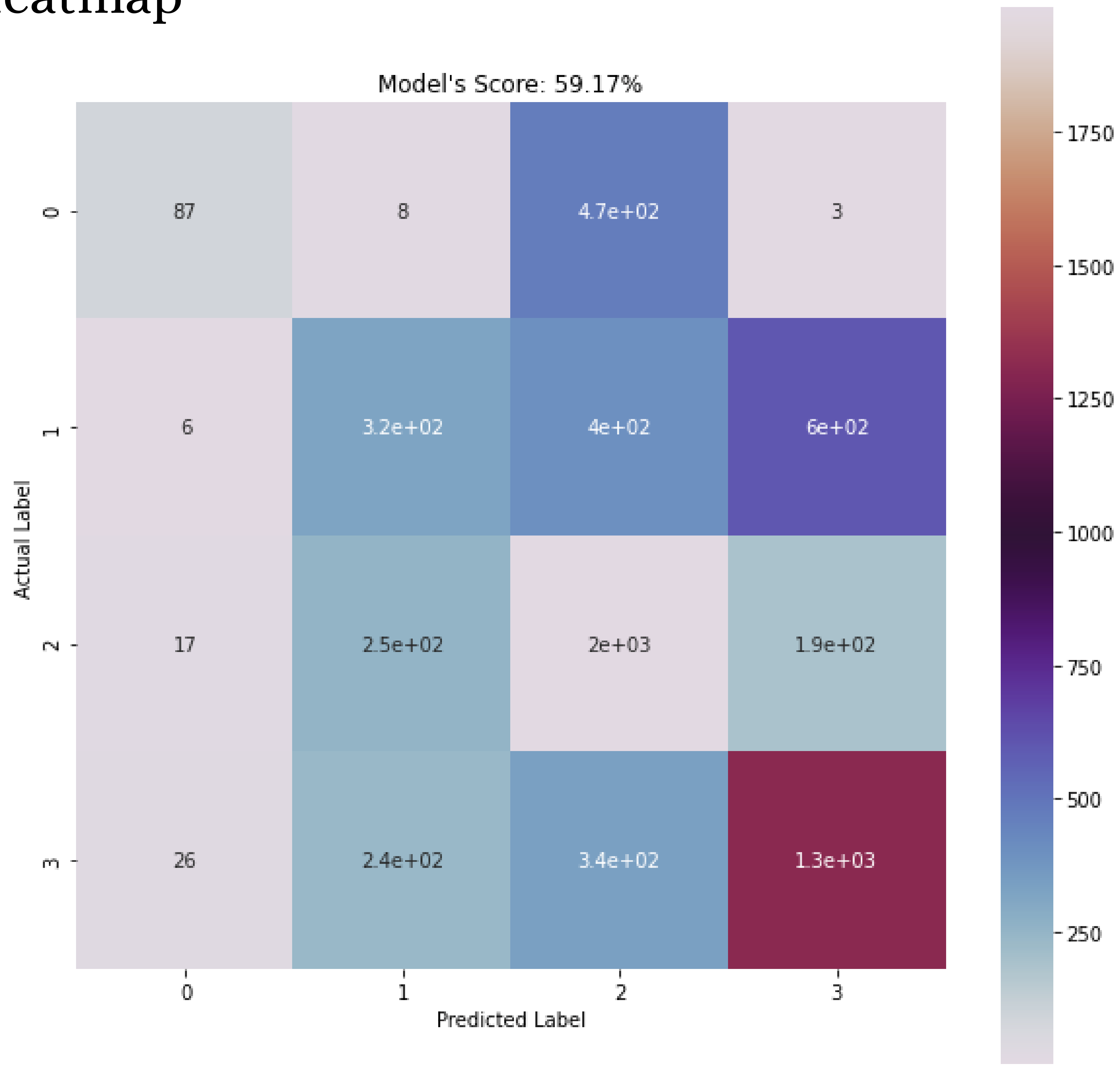
- Best Params

```
Best Criterion:  entropy
Best Max Depth:  9
Best Min Samples Leaf:  1
Best Min Samples Split:  3
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=9, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=3,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

- Print model's performance on different reports

```
Accuracy Score on Validation: 59.17%
Accuracy Score on Training: 47.77%



[[  87    8  472    3]
 [   6  324  402  601]
 [  17  252 1983  191]
 [  26  238  339 1308]]



              precision    recall  f1-score   support

           0       0.64      0.15      0.25       570
           1       0.39      0.24      0.30      1333
           2       0.62      0.81      0.70      2443
           3       0.62      0.68      0.65      1911

    accuracy                           0.59      6257
   macro avg       0.57      0.47      0.48      6257
weighted avg       0.57      0.59      0.56      6257
```

- Perform Model on Heatmap



Takeaways

- Using various Python Library and techniques for data preparation
- Understand Multi-Class Classification Problem (Library)
- Understand GridSearchCV
- Understand Pipeline
- Understand Feature Engineering
- Understand the pipeline of conducting Model Comparison and Model Evaluation

Drawbacks

- Low Score on Model Performance
- Does not understand some concepts or terminologies of ML Algorithm
- Features on Dataset may not be optimized

Thanks for watching