BDA 5012- Marketing Analytics
Project Report

- **Project Title:** Classification on Avila Dataset
- **Student Name:** Kutay Erkan
- **Dataset Name:** Avila Dataset

Abstract

Classification multi-class problems is an important problem of our age. When the classes are imbalanced, the problem becomes more complex; and also more intriguing. After feature engineering and oversampling of minority classes, Avila dataset is analyzed using 3 different models; Logistic Regression, AdaBoost, and Random Forests. Random Forests proves the be most accurate approach in the case of this problem.

Keywords: classification; class imbalance; avila; smote; random forests; logistic regression; adaboost

The Dataset

The Avila dataset can be found in UC Irvine Machine Learning depository. Dataset as it is consists of 10.430 observations, 10 features, and 1 target variable. Dataset doesn't have headers but they can be (and in our study, were) inserted using the paga of the dataset on UCIML.

What is the dataset about? By the exact words of the curators, it is:

"The Avila data set has been extracted from 800 images of the 'Avila Bible', an XII century giant Latin copy of the Bible."

Features in the dataset are characteristics of handwriting in the pages of this Bible, such as *intercolumnar distance, upper margin, row number,* and so on. All the values of the features are numerical and already standardized. The target variable is a categorical variable and denotes the copyist that has written that sample. As there are 12 different classes, this is a multi-class problem.

Exploratory Data Analysis & Feature Engineering

Firstly, a small sample of the dataset is inspected:

```
Sample data:
    intercolumnar_distance  upper_margin  lower_margin  exploitation  \
0                 0.266074     -0.165620      0.320980      0.483299
1                 0.130292      0.870736     -3.210528      0.062493
2                -0.116585      0.069915      0.068476     -0.783147
3                 0.031541      0.297600     -3.210528     -0.583590
4                 0.229043      0.807926     -0.052442      0.082634

   row_number  modular_ratio  interlinear_spacing    weight  peak_number  \
0    0.172340       0.273364             0.371178  0.929823     0.251173
1    0.261718       1.436060             1.465940  0.636203     0.282354
2    0.261718       0.439463            -0.081827 -0.888236    -0.123005
3   -0.721442      -0.307984             0.710932  1.051693     0.594169
4    0.261718       0.148790             0.635431  0.051062     0.032902

    m_r/i_s target
0  0.159345      A
1  0.515587      A
2  0.582939      A
3 -0.533994      A
4 -0.086652      F

Number of rows in the dataset: 10430
```

Figure 1: Sample data from Avila dataset

Other than our target variable, all of our features seem to be numerical. After that, we check minimum, maximum, standard deviation, and percentiles of our dataset; which are not shown here. Maximum values for most features appear incredibly high, especially with regard to

standard deviations of those features. Could there be an outlier? We check the boxplots. As the features are already standardized before this study, we can use same scale for all features.

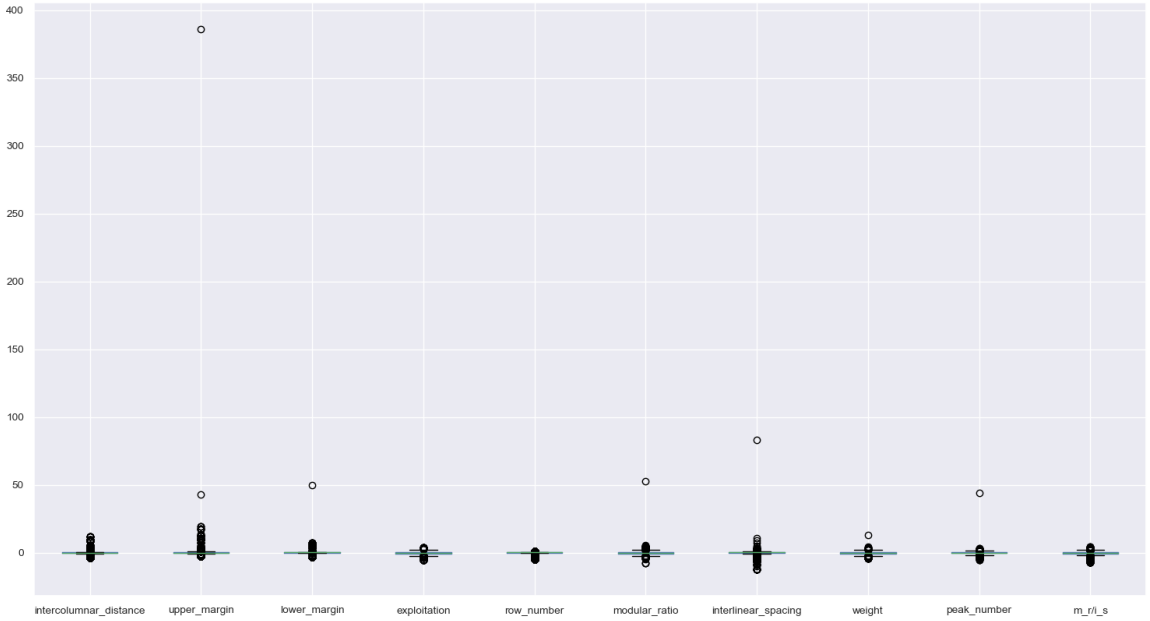


Figure 2: Boxplots of the features

There is indeed an outlier. We remove the single sample that has an upper_margin bigger than 350.

Next, we check the distribution of the sample between 12 different classes.

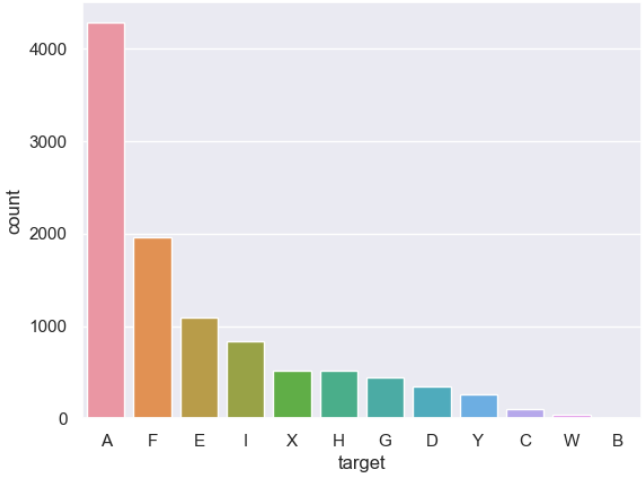


Figure 3: Distribution of samples to 12 different classes

There's a huge difference between the number of samples in different classes, for example, with the Class A and B. This is called **class imbalance problem**. It is generally mitigated with oversampling or undersampling methods. We will deal with this later on.

Next, we check the correlation matrix.

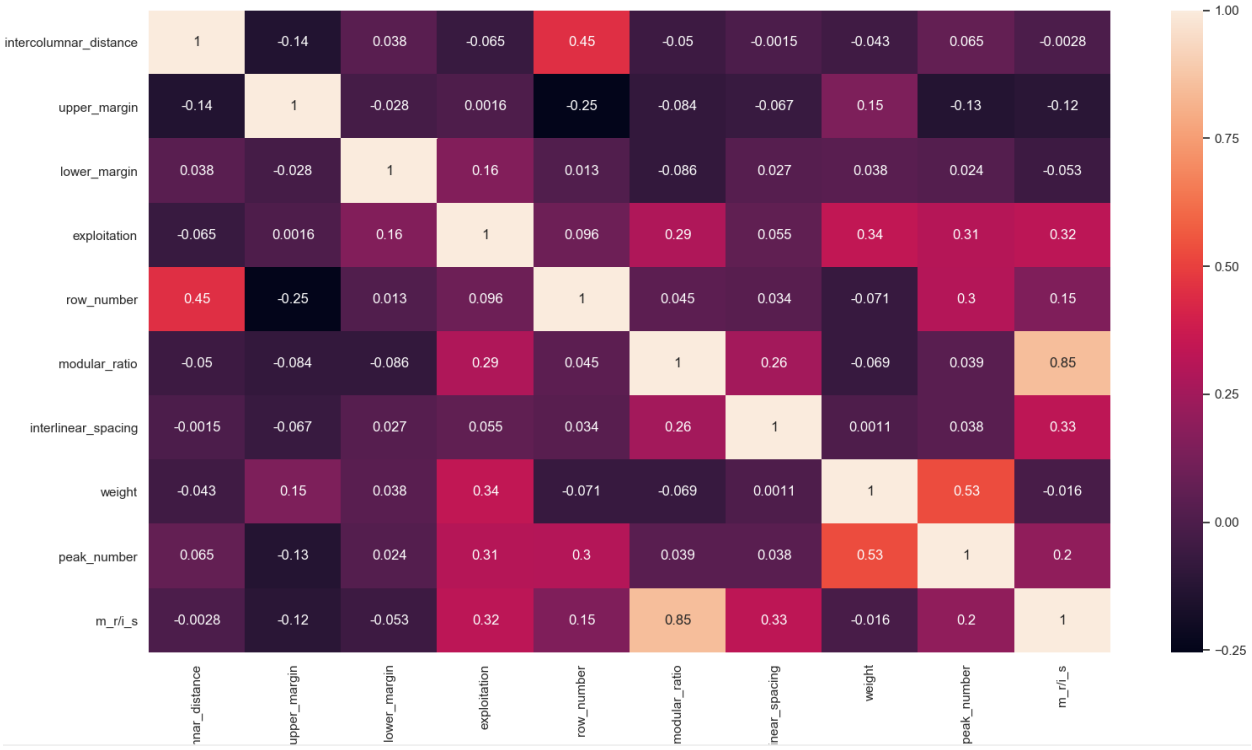


Figure 4: Correlation matrix of features

Genereally there is little correlation between features, which is desirable. One exception is correlation of $r^2 = 0.85$ between *modular_ratio* and *m_r/i_s*. As the latter feature is the ratio of *modular_ratio* and *interlinear_spacing*, hence the name *m_r/i_s*, it is logical that we have high correlation.

This correlation can also be seen in the pairplots of features. While most features seem unrelated, aforementioned features are highly correlated with little variance.

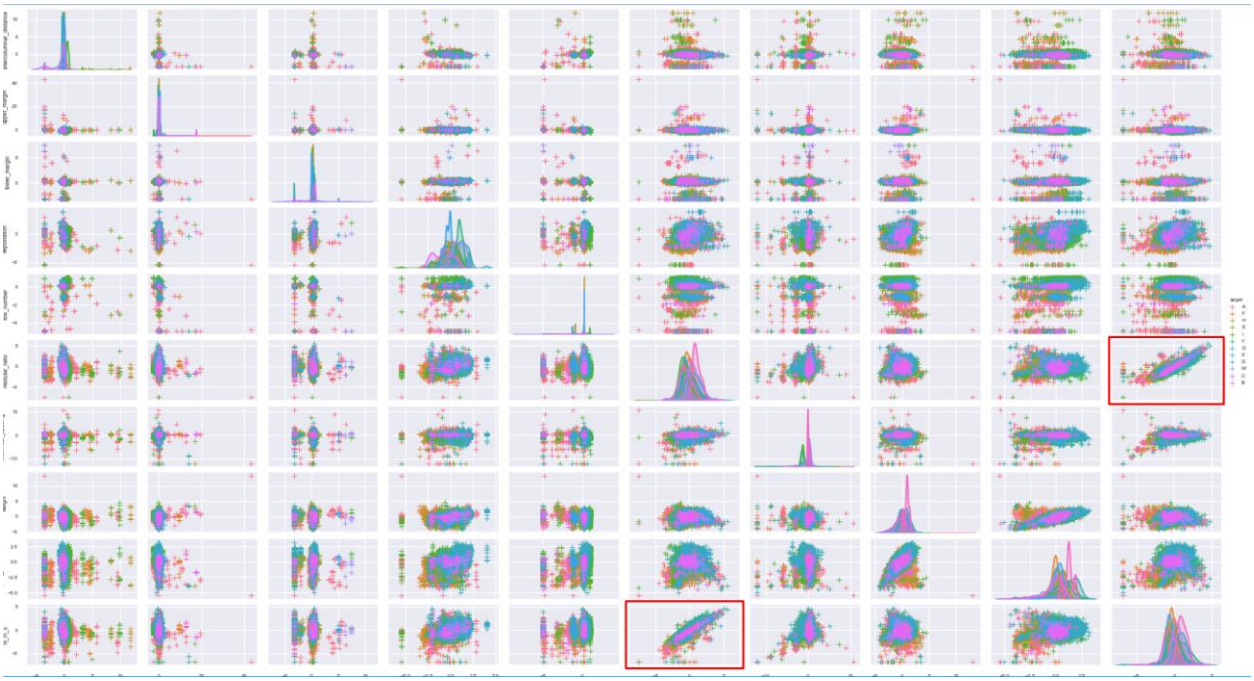


Figure 5: Pairplots of features, highly correlated features are circled in red.

Before we go further into the dataset, we separate our dataset to training and test datasets. This is because we will calculate Variance Inflation Factors, and we don't want data leakage from test dataset to training dataset. 30% of the dataset is held out, which is equal to 3.219 samples. We are going to be training our models on the 70% of the dataset, which is 7.300 samples. Stratified shuffle split is used as we have class imbalance.

After that, we calculate Variance Inflation Factors on our dataset. VIF = 4 is used as a threshold, as it equals to $r^2 = 0.87$ where $VIF = 1 / (1 - r^2)$, which is high enough that we are suspicious of **multicollinearity.** As we have previously suspected, *m_r/i_s* is dropped from the training dataset. We also drop this feature from the test dataset.

```
m_r/i_s is dropped with VIF=  4.22

Remaining features with VIF < 4:
intercolumnar_distance    1.29
upper_margin              1.15
lower_margin              1.05
exploitation              1.37
row_number                1.55
modular_ratio             1.26
interlinear_spacing       1.09
weight                    1.74
peak_number               1.74
```

Figure 6: VIFs of different features after removing *m_r/i_s*

We had talked about class imbalance problem. To mitigate this, SMOTE (Synthetic Minority Over-sampling Technique) is applied on the training dataset. New, synthetic samples are created to make number of samples equal among all classes. As such, now training dataset consists of 35.988 samples.

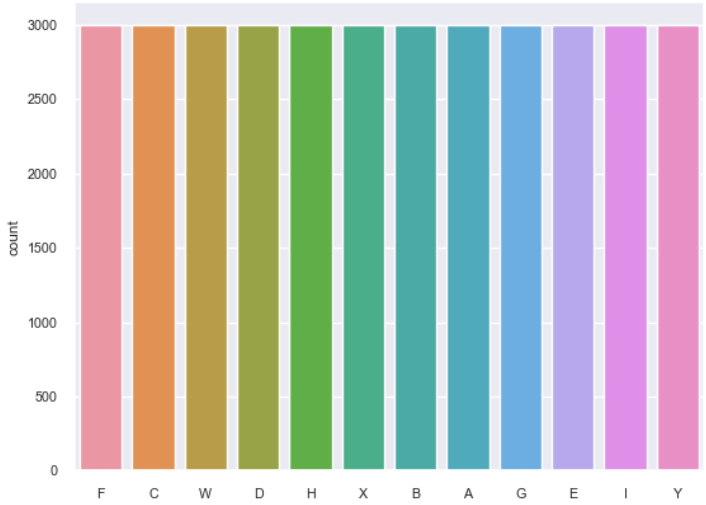


Figure 7: Distribution of samples to classes after SMOTE

Model Selection

Three different models are tried and tested. For each algorithm, **3-fold Cross Validation** is applied. Scoring metric is **Accuracy**, which is shown with **Mean ± 2 * Standard Deviation of Accuracy**. The reason for this is presenting accuracy with **95% Confidence Interval**. Confusion matrices for different models are also presented.

First algorithm that was tried is **Logistic Regression**. Firstly, the following grid search is carried out:

- C: [0.001, 0.01, 0.1, 1, 10, 100]
- solver: ['newton-cg', 'sag', 'saga', 'lbfgs']

The selected hyperparameters are C: 100 and solver: newton-cg. We input these hyperparameters to our Logistic Regression model. The result is below:

Accuracy: 0.574 (+/- 0.017 with 95% CI)

Accuracy: 0.574 (+/- 0.017)

Predicted	A	B	C	D	E	F	G	H	I	W	X
Actual											
A	817	0	221	298	170	537	389	122	56	254	43
B	0	2999	0	0	0	0	0	0	0	0	0
C	35	0	1128	92	313	2	131	1083	28	150	7
D	346	0	223	958	396	110	146	405	120	141	7
E	115	0	271	161	830	164	308	378	23	208	346
F	506	0	217	208	103	855	500	403	44	110	0
G	214	0	120	6	22	176	1403	814	0	140	59
H	96	0	156	12	278	147	601	1533	69	5	48
I	23	0	72	0	5	38	9	6	2624	49	65
W	0	0	0	0	0	0	0	0	0	2999	0
X	10	0	7	37	12	5	13	121	64	99	2317
Y	42	0	21	44	50	3	39	79	115	228	185
All	2204	2999	2436	1816	2179	2037	3539	4944	3143	4383	3077

Predicted	Y	All
Actual		
A	92	2999
B	0	2999
C	30	2999
D	147	2999
E	195	2999
F	53	2999
G	45	2999
H	54	2999
I	108	2999
W	0	2999
X	314	2999
Y	2193	2999
All	3231	35988

Figure 8: Logistic Regression with 3-fold CV

Next algorithm is AdaBoost. With 50 estimators and a learning rate of 1, AdaBoost gives the following results:

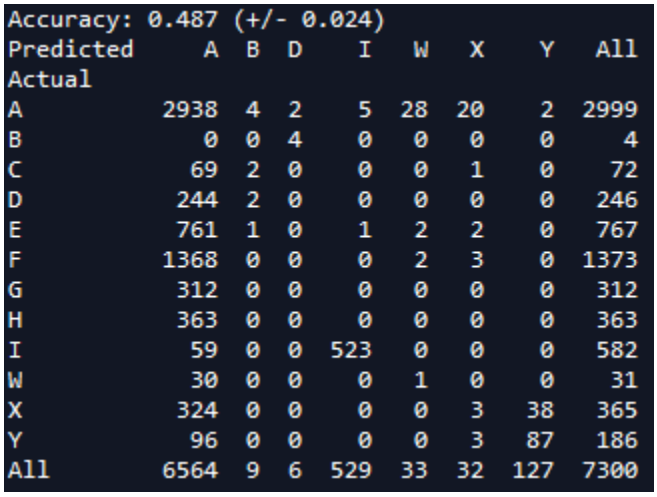
Accuracy: 0.279 (+/- 0.044 with 95% CI)

Accuracy: 0.279 (+/- 0.044)

Predicted	A	B	C	D	E	F	G	H	I	W	X	Y	All
Actual													
A	158	1	1	70	0	3	2271	0	26	328	121	20	2999
B	2100	0	300	599	0	0	0	0	0	0	0	0	2999
C	61	7	19	85	9	0	2364	0	5	350	97	2	2999
D	7	2	3	61	0	0	2504	0	5	335	73	9	2999
E	28	0	0	41	0	0	2511	0	1	318	92	8	2999
F	66	0	0	20	0	1	2529	1	7	302	62	11	2999
G	0	0	0	0	0	0	2580	0	0	354	61	4	2999
H	54	0	0	10	0	3	2523	5	32	280	83	9	2999
I	104	0	0	39	0	0	0	0	2702	8	23	123	2999
W	110	0	0	858	0	0	250	0	0	1598	183	0	2999
X	9	0	0	26	0	0	1466	0	8	496	539	455	2999
Y	18	0	0	54	0	0	23	0	66	15	438	2385	2999
All	2715	10	323	1863	9	7	19021	6	2852	4384	1772	3026	35988

AdaBoost yields a really low accuracy on our traning dataset. Without SMOTE, our accuracy improves, but AdaBoost doesn't estimate for minority classes:

Accuracy: 0.487 (+/- 0.024 with 95% CI)



Accuracy: 0.487 (+/- 0.024)

Predicted	A	B	D	I	W	X	Y	All
Actual								
A	2938	4	2	5	28	20	2	2999
B	0	0	4	0	0	0	0	4
C	69	2	0	0	0	1	0	72
D	244	2	0	0	0	0	0	246
E	761	1	0	1	2	2	0	767
F	1368	0	0	0	2	3	0	1373
G	312	0	0	0	0	0	0	312
H	363	0	0	0	0	0	0	363
I	59	0	0	523	0	0	0	582
W	30	0	0	0	1	0	0	31
X	324	0	0	0	0	3	38	365
Y	96	0	0	0	0	3	87	186
All	6564	9	6	529	33	32	127	7300

Figure 10: AdaBoost with 3-fold CV, no oversampling (SMOTE)

Final algorithm that we use is Random Forests. With 100 estimators and entropy as the criterion, Random Forests performs really well.

Accuracy: 0.997 (+/- 0.004 with 95% CI)

Accuracy: 0.997 (+/- 0.004)

Predicted	A	B	C	D	E	F	G	H	I	W	X
Actual											
A	2985	0	0	2	2	3	2	5	0	0	0
B	0	2999	0	0	0	0	0	0	0	0	0
C	0	0	2999	0	0	0	0	0	0	0	0
D	0	0	0	2999	0	0	0	0	0	0	0
E	7	0	0	4	2985	1	0	1	0	0	1
F	27	0	0	0	3	2937	21	11	0	0	0
G	1	0	0	0	1	1	2996	0	0	0	0
H	0	0	1	0	3	1	3	2991	0	0	0
I	0	0	0	1	0	0	0	0	2998	0	0
W	0	0	0	0	0	0	0	0	0	2999	0
X	0	0	0	0	6	0	0	0	0	0	2988
Y	0	0	0	1	1	0	0	0	0	0	5
All	3020	2999	3000	3007	3001	2943	3022	3008	2998	2999	2994

Predicted	Y	All
Actual		
A	0	2999
B	0	2999
C	0	2999
D	0	2999
E	0	2999
F	0	2999
G	0	2999
H	0	2999
I	0	2999
W	0	2999
X	5	2999
Y	2992	2999
All	2997	35988

Figure 11: Random Forests with 3-fold CV

It is clear Random Forests is the clear winner. It also performs well on our test dataset, 30% of the initial dataset:

Accuracy: 0.957 (+/- 0.024 with 95% CI)

```
Accuracy: 0.957 (+/- 0.024)
```

Predicted	A	C	D	E	F	G	H	I	W	X	Y	All
Actual												
A	1272	0	0	6	6	2	0	0	0	0	0	1286
B	0	0	0	0	0	1	0	0	0	0	0	1
C	6	24	0	0	0	0	1	0	0	0	0	31
D	1	0	104	1	0	0	0	0	0	0	0	106
E	16	0	4	302	3	0	0	0	0	3	0	328
F	45	0	0	0	537	5	1	0	0	0	0	588
G	7	0	0	0	0	127	0	0	0	0	0	134
H	6	1	0	0	2	0	147	0	0	0	0	156
I	0	0	0	1	0	0	0	248	0	0	0	249
W	1	0	0	1	0	0	0	0	11	0	0	13
X	3	0	0	4	0	0	0	0	0	148	2	157
Y	3	0	0	0	0	0	0	0	0	2	75	80
All	1360	25	108	315	548	135	149	248	11	153	77	3129

Figure 12: Random Forests with 3-fold CV on test dataset

We know that Random Forests is the most optimal model for our problem. Which features are the most important ones in our model? We can check feature importances to see this:

```
Features sorted by their score:
[(0.1928, 'intercolumnar_distance'), (0.1794,
'row_number'), (0.1786, 'upper_margin'), (0.1491,
'exploitation'), (0.1452, 'lower_margin'), (0.0662,
'peak_number'), (0.0504, 'interlinear_spacing'), (0.0235,
'modular_ratio'), (0.0148, 'weight')]
```

Figure 13: Feature importances of the final model

Using the most important features, we can see classes are almost differentiable to human eye:

Figure 14: Plot of upper_margin and intercolumnar_distance

Conclusion

As Avila dataset is already curated, minimal data preprocessing was needed. It can be said Avila dataset is a relatively simple and enjoyable dataset for those who want to learn more about multi-class classification problems with class imbalance. Random Forests proved to be the best option in this study. For further research, analyzing why AdaBoost performed so poorly might be interesting.

References

1) https://archive.ics.uci.edu/ml/datasets/Avila
2) https://chrisalbon.com/machine_learning/trees_and_forests/adaboost_classifier/
3) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
4) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
5) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html