BOOSTING WITH ORIGINAL AND CLUSTERED CATEGORICAL FEATURES FOR MACHINE LEARNING ON LARGE DATASETS

Giovanni Bellio

Auburn University at Montgomery

2022

BOOSTING WITH ORIGINAL AND CLUSTERED CATEGORICAL FEATURES FOR MACHINE LEARNING ON LARGE DATASETS

by

Giovanni Bellio

A thesis submitted to the Graduate Faculty of
Auburn University at Montgomery
in partial fulfillment of the
requirements for the Degree of
Master of Science

in

Computer Science

Montgomery, Alabama
29 July 2022

Approved by

Olcay Kursun Digitally signed by Olcay Kursun Date: 2022.07.27 23:12:51 -05'00'
Dr. Olcay Kursun
Thesis Director

Randy D. Russell Digitally signed by Randy D. Russell Date: 2022.07.28 19:10:24 -05'00'
Randy Russell
Thesis Co-Director

Zhimin Gao Digitally signed by Zhimin Gao Date: 2022.07.28 13:25:16 +08'00'
Dr. Zhimin Gao
First Reader

Lei Wu Digitally signed by Lei Wu Date: 2022.07.28 15:08:45 -05'00'
Dr. Lei Wu
Second Reader

Matthew Ragland Digitally signed by Matthew Ragland Date: 2022.07.29 11:18:38 -05'00'
Dr. Matthew Ragland
Associate Provost

ACKNOWLEDGMENTS

I want to express my deepest appreciation to my thesis advisor, Dr. Olcay Kursun, Department of Computer Science at Auburn University at Montgomery. He was always available and supportive, willing to teach new concepts and algorithms needed for this thesis study every time we met.

Dr. Kursun provided me with some of the datasets, computer programs and algorithms as part of the NSF grant 2003740 on Hyperspectral Cloud Segmentation together with Professors Randy Russell, Luis Cueva Parra, and Semih Dinc.

This work was supported, in part, by the National Science Foundation under Grant No. 2003740.

I would also like to extend my gratitude to all the faculty at Auburn University at Montgomery. Their teaching throughout the completion of my Undergraduate and Graduate degrees was fundamental for my academic development, which inspired me to continue learning and pursuing a career in the area of Computer Science. I would like to specially thank Dr. Lei Wu, Dr. Zhimin Gao, and Dr. Semih Dinc for all their guidance during these years at AUM.

Finally, I would like to thank my family for all their endless support during my life. Everything I have achieved has been possible because of them, so for that I will be forever grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Compared to developing single models, ensemble learning algorithms that utilize decision trees (DTs) and boosting have received increasing interest due to many features including but not limited to their fast and accurate predictions, robustness to noise, ability to deal with diverse features such as both numerical and categorical features, having fewer parameters to optimize, and having a rule-based interpretability using if-then-like rules. Gradient/Adaptive boosting methods based on decision trees, such as CatBoost and AdaBoost, can handle diverse data types and solve a wide range of machine learning problems involving categorical variables. The thrust of this thesis is to develop machine learning algorithms based on such boosting algorithms and test their applicability and prediction performances on several datasets. These algorithms are showcased in this thesis with two types of data. One is the Movie Ratings dataset and the other one is three Hyperspectral Image (HSI) datasets for pixel classification. These two types of data consist of very different characteristics.

Movie Ratings dataset is composed of both numerical and categorical features (such as genre and actors) and requires a regression machine learning model to predict the ratings. Moreover, some of the categorical features in the Movie Ratings dataset have high cardinality (having a large number of categories, such as the main actor of the movie).

On the other hand, in the Hyperspectral Image (HSI) domain, there are high number of numerical features and a classification machine learning model is required to predict classes of pixels. There are vegetation vs. soil types of classes in two of the HSI

datasets used and there are cloud-classes (e.g., dark vs. thin clouds) in the third HSI dataset. Interestingly, HSI dataset does not contain any categorical features but the thesis proposes and demonstrates that creation of categorical features using clustering algorithms proves very useful in enriching the data representation for categorical-boosting ensembles.

For all the datasets used in this thesis, gradient boosting methods performed favorably to the benchmark algorithms. This thesis presents a method that can be further developed for achieving dimensionality reduction, high-accuracy classification, and implementation in high-performance computing frameworks for hyperspectral image classification.

CHAPTER 1: INTRODUCTION

As the performance of machine learning (ML) models perform differently on different dataset, there is no guaranteed single best off-the-shelf machine learning algorithm for a given learning task/dataset, which is also known as "No Free Lunch Theorem" in machine learning (Alpaydin 2014). Nevertheless, ensemble methods are known to achieve low variance; that is, compared to using a single model, an ensemble method has smaller deviation in the learned functions from one training run to another. Boosting is a meta-learning algorithm and it can combine classifiers to create powerful ensembles. Popular implementations of boosting such as AdaBoost, XGBoost, and CatBoost uses decision trees (shallow decision trees or decision tree stumps) and compared to single ML models. Decision trees are data structures that repeatedly divide the dataset into smaller subsets based on applying thresholds on features with the goal of minimizing impurity (until maximum depth is reached or one class left in the branch/leaf). Boosting methods enjoy robustness to noise, ability to deal with both numerical and categorical features, having fewer parameters to optimize, and having a rule-based interpretability using if-then-like rules (Samat et al. 2021; Fernandez-Delgado et al. 2014).

In this thesis, boosting algorithms are studied and their feasibility is tested on some challenging datasets. One of the datasets is the Movie Ratings dataset, which is composed of both numerical (such as year and duration of the movie) and categorical features (such as genre and actor names). It requires a regression model for predicting the movie ratings from these features. Moreover, there are a variety of genres available in the categorical genre feature but the cardinality of the actor feature is much greater. There are

a much larger variety of distinct categories for the actor-related fields; each actor name is a different category and there are many actors in the dataset.

The other datasets are from the field of Hyperspectral Imaging (HSI). The HSI datasets are composed of large HSI images with the task of classification of individual pixels. Typically, for every HSI image a ground truth image is provided that contains the class labels of individual pixels. An exemplary application could be classification of individual pixels into various types of vegetation/soils/fields/trees, and another application could be classification of individual pixels into dark, regular, or thin clouds versus clear sky. In HSI datasets, there are high number of numerical features and a classification machine learning model is required. Typically, HSI datasets contain several hundred of numerical features (each one corresponding to reflectivity/irradiances in a different wavelength) but they do not contain any categorical features. However, boosting methods can take advantage of a preprocessing proposed in this thesis to create new categorical features using clustering algorithms for enriching the data representation, and thus increasing the classification accuracy and reducing dimensionality.

CatBoost was the first boosting algorithm used in this thesis. In the early phases of the thesis, CatBoost was identified to be an efficient method needed to deal with the categorical features available faced in predicting the movie ratings. “CatBoost” name comes from two words “Category” and “Boosting”. CatBoost was developed by Yandex and released to the open-source community in 2017 and became popular primarily due to its ability for handling categorical features without overfitting. It can also be trained very fast on GPU. For this thesis, CatBoost’s Python programming language implementation was used with the following easy set up command: “pip install catboost”. As described

above, in this thesis, CatBoost was tested with different datasets. The first task at hand deals with movie ratings prediction using the Movie Rating dataset (Section 2.1), which contains several categorical fields that were directly extracted from IMDb's official website. This work proposes the idea of predicting ratings for movies before their release, which is a feature yet to release on current applications. To achieve this, several approaches are proposed (Section 3.1), where each of these use the categorical data with different methods in order to calculate accurate prediction values results with a high coefficient of correlation (Section 4.1) using CatBoost as its model regressor.

The second task at hand refers to the prediction of pixel values of hyperspectral images. In Hyperspectral Imaging (HSI), a pixel is characterized by a high number of spectral channels/bands, thus allowing accurate and efficient classification of individual pixels (Sellami et al. 2019; Grana et al. 2018; GISGeography 2022; Kursun et al. 2021). HSI cameras vary in the number of wavelengths (bands) they have, but typically in an HSI dataset, every pixel is represented by several hundreds of bands. The spectral signature in those bands (reflectivity/irradiances in different wavelengths) for a pixel can be used as a powerful predictor of the class-label (i.e., for classification of that individual pixel). Since different classes have different certain hyperspectral signatures, HSI can serve as an important pattern recognition goal, for example, for scanning a large field by taking aerial pictures. In such an HSI application, single pixel classification can help monitor the state of crops (wet/dry/rotten) or to find irregularities such as a metal object camouflaged in the field. This thesis proposes a categorical-boosting-classification method that utilizes CatBoost (Section 3.2) as a viable solution to the pixel classification problem. This dataset is currently being extracted by the AUM-Hyperspectral team

(Kursun et al. 2022). Therefore, it was not possible to test the proposed algorithm on the finalized dataset. Consequently, to overcome this inconvenience, this work used two benchmark HSI datasets, Indian Pines and Salinas, as presented in Section 2.2. The experimental results are presented in Section 4.2.

Moreover, Hyperspectral imaging can potentially be of great benefit for modern applications. In the case of hyperspectral sky imaging (such as the AUM-Hyperspectral team's dataset), a potential application is the development of an automated ground-based system for detecting the amount and type of cloudiness. Such systems could improve the spatial and temporal resolution of cloud information vital to understanding Earth's climate. In the case of hyperspectral imaging of the Earth's surface (such as the Indian Pines and Salinas datasets) the images yield important information on land use and agricultural productivity.

The thesis is organized as follows. The datasets, including data extraction and data statistics, is described in Chapter 2. The proposed machine learning models using categorical boosting are described in Chapter 3. The experimental results on the datasets are presented in Chapter 4. Finally, the conclusions and future work are discussed in Chapter 5.

CHAPTER 2: DATA

This chapter provides a thorough explanation on how the data used in this thesis was obtained, and what each of the feature columns in the datasets represent. Two sections will be discussed in this chapter: The Movie Ratings dataset on Section 2.1, and the Hyperspectral Imaging datasets on Section 2.2. Every dataset will be converted to two tables: table X and table Y. In machine learning the goal is to predict values from table Y after training the model with values from table X (Pedrosa et al. 2011). An example of this is visualized on Figure 2.1 where the feature matrix (X) is a table where there are D number of columns and N number of rows from the dataset, and the target vector (Y) which is a table with a single column (not present on table X) with N number of rows.

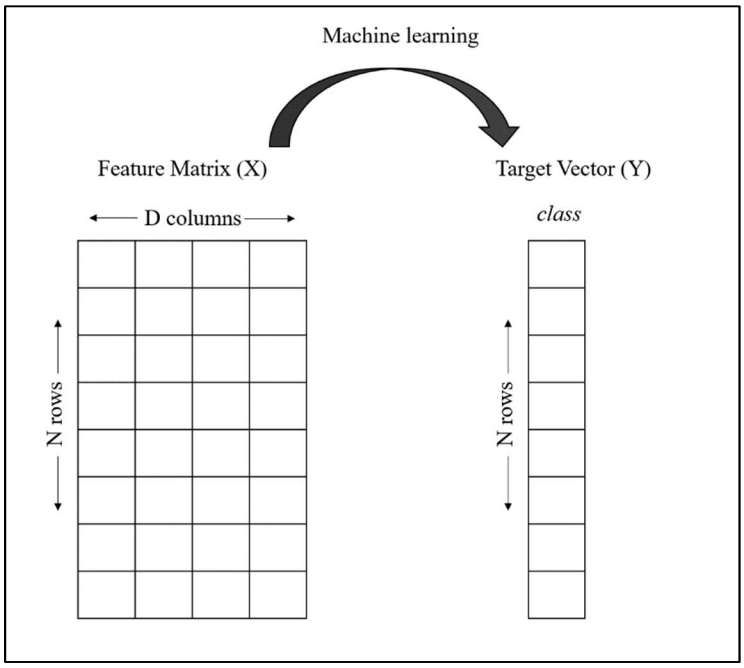


Figure 2.1. Visualization of mapping/learning task of machine learning as a function from table X (the training dataset with N examples and D features) to table Y (the class-labels of those N training examples)

2.1 MOVIE RATINGS

This dataset consisting of 21,508 sentences and 14 fields of mainly categorical data describes information about several movies released from 1912 until 2021. The objective of this work was to utilize columns of said dataset, such as actors and genres, to train the CatBoost machine learning algorithm and be able to predict movie ratings as accurately as possible (Gulin et al. 2018).

2.1.1 DATA EXTRACTION

The data was obtained from the International Movie Database official's website (IMDb) where there is available data free of access ("IMDb datasets", 2021). Several TSV (tab separated format) files can be downloaded by the public in this site, where the following tsv files: title_basics, title_principals, title_ratings, and name_basics were obtained to start the creation the Movie Ratings dataset. After the files were downloaded, the next step was preprocessing the data to have it machine-learning-ready for our program, which meant that a lot of irrelevant data had to be erased. For every file, each column was considered for keeping in the final dataset, but only those that seemed more relevant for predicting ratings of movies were kept. For example, some fields that were not considered were: titleType, originalTitle, isAdult, endYear from the title_basics file. On another file called title_principals, some fields that were also unconsidered were: category, job, and characters. Lastly, on the name_basics file, the removed fields were: birthYear, deathYear, primaryProfession, and knownForTitles. After manually deleting the unnecessary columns, all the data that was needed was available, but it was separated and spread across four different files. Therefore, to only obtain the final table

result in one file, a database was created in order to store and combine the data on all of the files.

The data from each of the files was inserted into tables in a database (Oracle MySQLWorkBench 2021), and after some select commands were implemented, it was possible to obtain queries that would contain the desired final data. The query result that consisted of everything that was needed was a table which was imported into a Python program in a CSV (comma separated format) file (Van Rossum et al. 2009). This table contained 695,974 rows and 10 columns, which consisted of title_id, rating, numVotes, title_name, year, genres, person_id, person_job, person_name, and runtime.

Although having this large amount of data was promising, it was necessary to do some further processing since some movies had missing values in all the genres and year of release columns. Therefore, every movie that had these missing values was discarded. This resulted on the remaining movies having a high number of votes, which proved that movies that were more popular had more reliable data. On top of this, various column values in the table were re-arranged so that the data could later be processed by algorithms in a more effective way. An example of this was making sure that each movie would appear in only one row, instead of multiple times. In the original table, for every different person that participated in a movie, a whole new row was needed which would lead into the same movie appearing in multiple rows. Therefore, finding a way to fit every single person that worked in a movie in just one row was an important step in the setup of the dataset.

The concluding dataset consisted of 21,508 rows and the following 14 columns: title_id, title_name, rating, year, numVotes, genre1, genre2, genre3, actor1, actor2, actor3, actor4, director_name, and runtime. From this dataset, it was possible to obtain the table X and table Y that were used for the machine learning section (Section 3) of this work. Table X consisted of 21,508 rows and 10 columns which were: year, genre1, genre2, genre3, actor1, actor2, actor3, actor4, director, runtime. Table Y consisted of 21,508 rows and 1 column which was the rating column.

2.1.2 DATA STATISTICS

With a final dataset table ready to implement into our program, it was important to understand the contents of it regarding every column and row. To start, ratings were separated from the rest of the columns since those were the values that the model was meant to predict. These were stored in a separate table called Y, and the rest of the data was stored in a table called X (as described in Figure 2.1).

Regarding table Y, the data contained float values that represented the rating of the movie from a range of 1 through 10. Figure 2.2 demonstrates a histogram with the number of movies per rating in the ratings column (Hunter 2007).

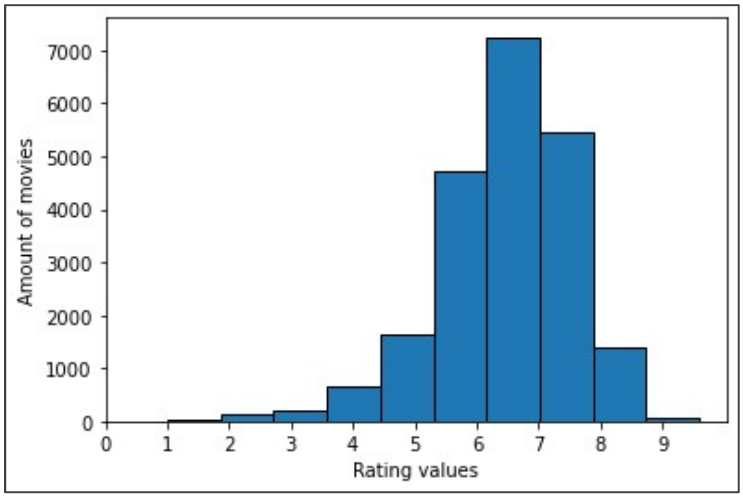


Figure 2.2. Movies per ratings in the ratings column

On table X, there were a total of 13 columns, which were: title_id, title_name, year, numVotes, genre1, genre2, genre3, actor1, actor2, actor3, actor4, director_name, and runtime. First, the column “year” contained the release year of a movie released between 1912 and 2021. A histogram that displays the distribution of these movies by their release date is shown on Figure 2.3.

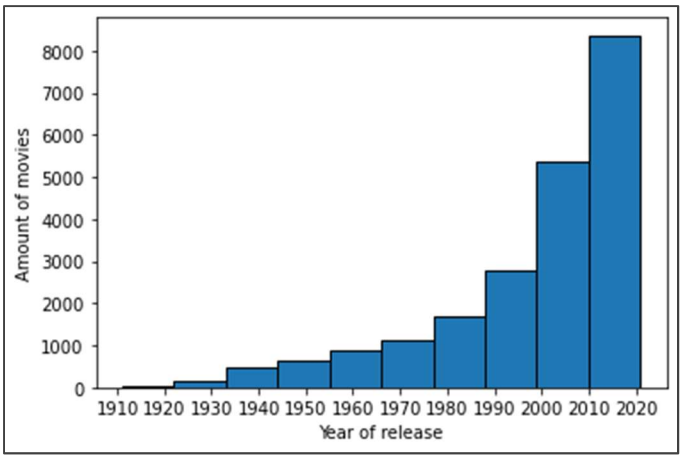


Figure 2.3. Distribution by release date in the year column

As it can be seen on Figure 2.2, there were movies stored from every single decade since the first release in 1911. As dates get more recent, the more data available there is. It was very beneficial to have many movies from 2010's decade, since the goal of this thesis is to predict movies that will be released in the upcoming years after 2020. The reason for this is that films from the last decade have the same viewing audience and many similar features than movies that will be released in years to come.

As for other columns, the actors were separated in the four columns: actor1, actor2, actor3, and actor4. In one hand, the actor1 contained the principal actor or actress of a movie. In the other hand, actor 2, actor3, and actor4 were actors with a secondary role of a movie, but still an important part of the cast. There were some cases where some movies would not contain more than one or two actors, so a '0' was inserted for the missing values in that specific row. Figures 2.4, 2.5, 2.6, and 2.7 show the count of the top 25 results for each of the actor columns respectively.

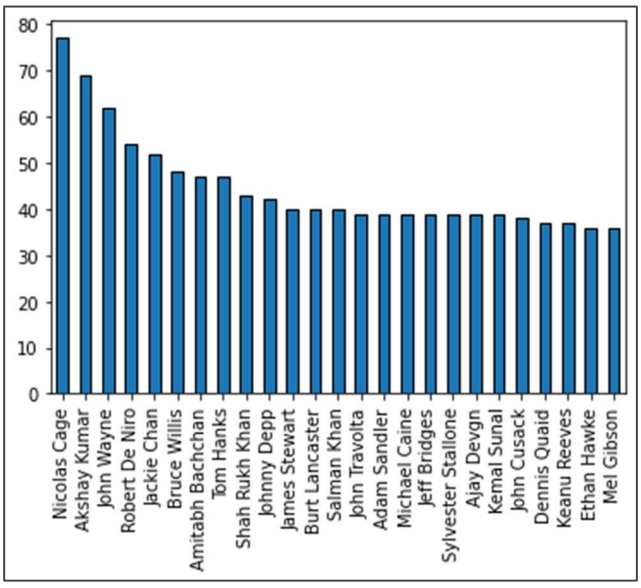


Figure 2.4. Top 25 actors count in the actor1 column

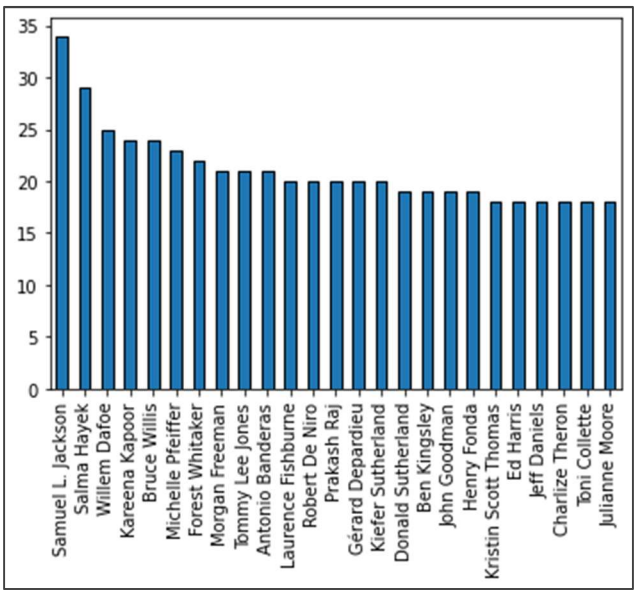


Figure 2.5. Top 25 actors count in the actor2 column

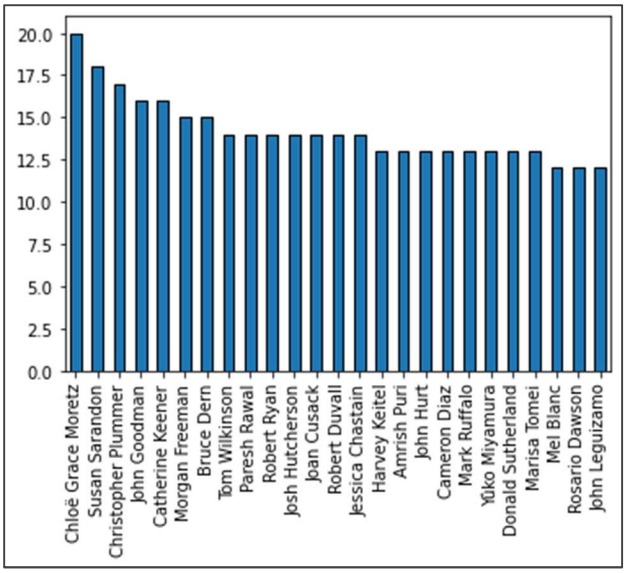


Figure 2.6. Top 25 actors count in the actor3 column

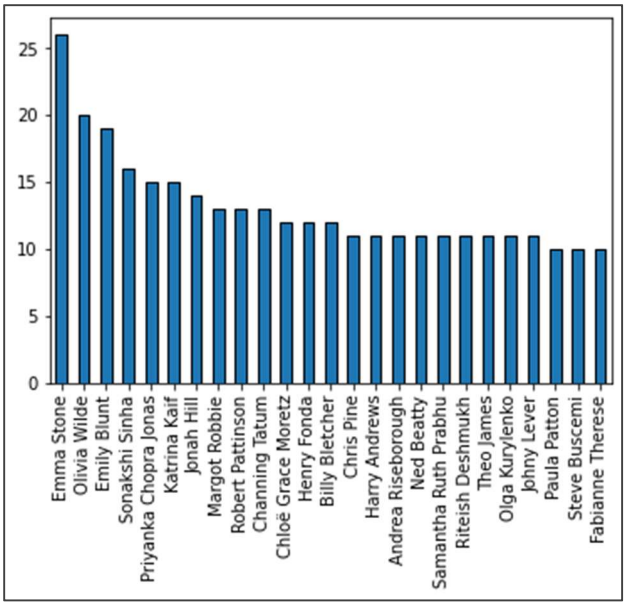


Figure 2.7. Top 25 actors count in the actor4 column

Some of other columns present on table X were genre1, genre2 and genre3. The first one of these represented the most relevant genre to categorize the movie. The other two were situational because if a movie only contained one genre instead of two or three, a 0 was inserted in these remaining columns to represent that the movie had no more genres. Figures 2.8, 2.9, and 2.10 demonstrate the distribution (count) of each of the genres respectively with bar plots.

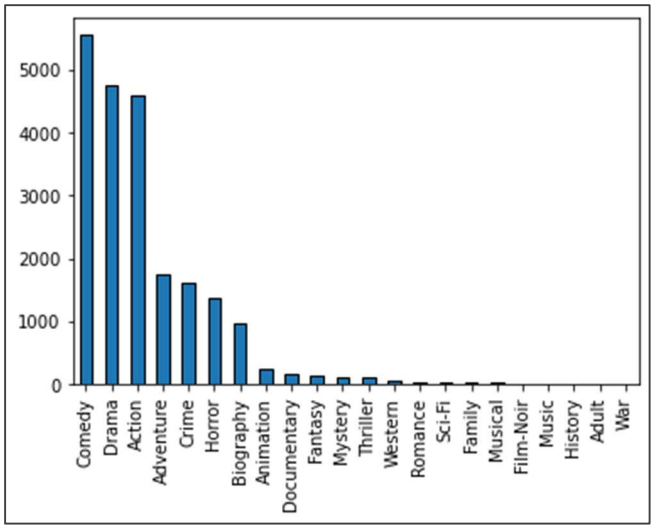


Figure 2.8. Distribution of genres in the genre1 column

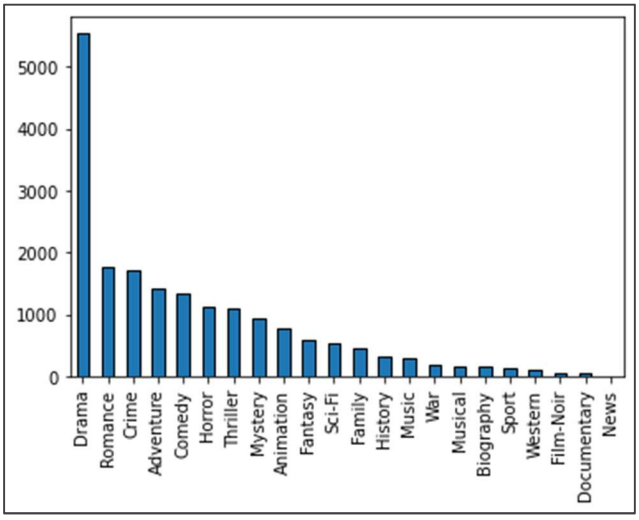


Figure 2.9. Distribution of genres in the genre2 column

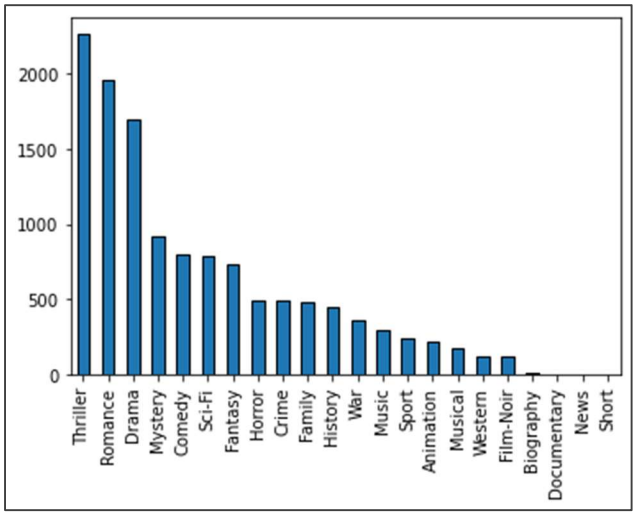


Figure 2.10. Distribution of genres in the genre3 column

Finally, the last two columns in the table to discuss are the director_name and runtime columns. They contained the director's name and the length in minutes of the movies respectively. Figure 2.11 shows the count of directors for the first 25 results, and Figure 2.12 displays the demonstrates a distribution of length in minutes in the runtime column.

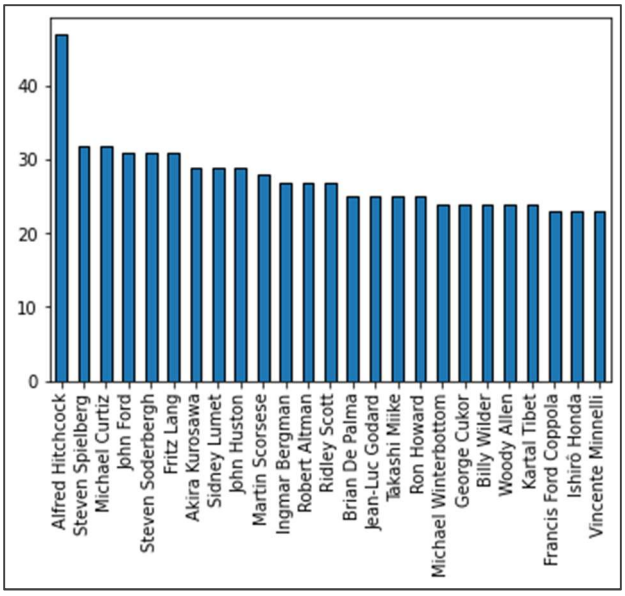


Figure 2.11. Top 25 directors count in the director_name column

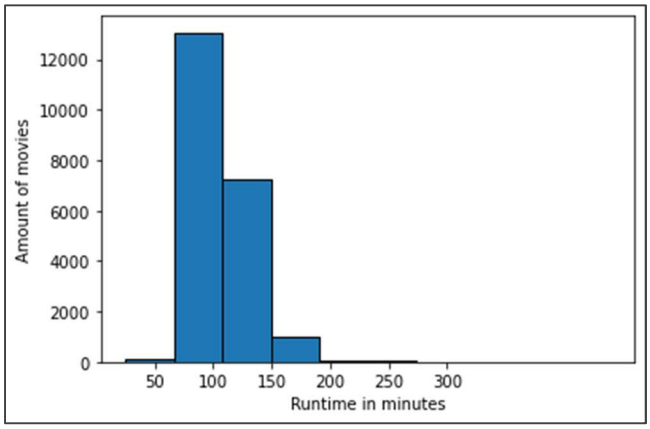


Figure 2.12. Distribution of length in minutes in the runtime column

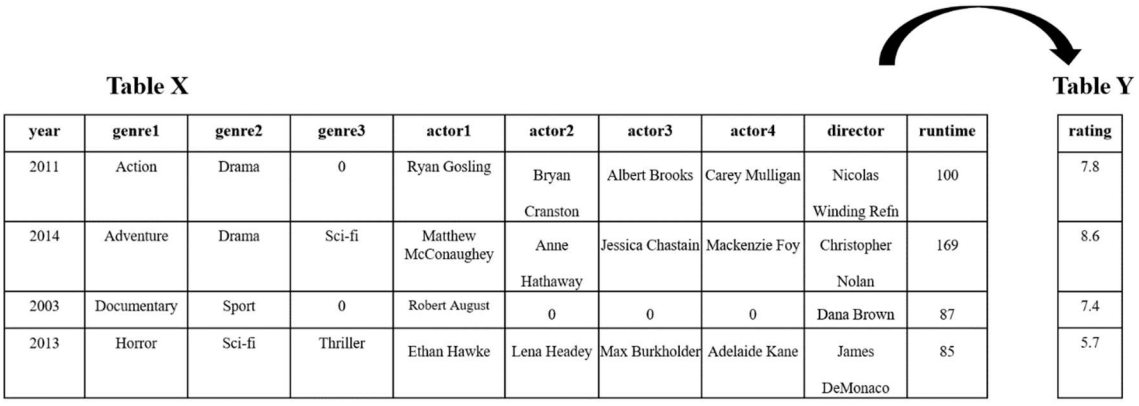
Other columns from the dataset such as: title_id, title_name, numVotes, had to be dropped at the moment of applying the machine learning algorithm. The title_id and title_name columns because they were used for only identification purposes, and the numVotes column because a new movie will never have any votes before release, so it

would not make sense to train the model with it included. Table 2.1 displays the initial table for the Movie Ratings dataset.

title_id	name	rating	year	votes	genre1	genre2	genre3	actor1	actor2	actor3	actor4	director	runtime
tt0780504	Drive	7.8	2011	599536	Action	Drama	0	Ryan Gosling	Bryan Cranston	Albert Brooks	Carey Mulligan	Nicolas Winding Refn	100
tt0816692	Interstellar	8.6	2014	1637059	Adventure	Drama	Sci-fi	Matthew McConaughey	Anne Hathaway	Jessica Chastain	Mackenzie Foy	Christopher Nolan	169
tt0308508	Step Into Liquid	7.4	2003	2668	Documentary	Sport	0	Robert August	0	0	0	Dana Brown	87
tt2184339	The Purge	5.7	2013	214294	Horror	Sci-fi	Thriller	Ethan Hawke	Lena Headey	Max Burkholder	Adelaide Kane	James DeMonaco	85

Table 2.1. Initial table of 14 columns of the movie ratings dataset

Then, after preparing the table for the machine learning stage, the table was modified to look like Figure 2.13, where ratings (Table Y) were separated from the rest of the columns (Table X) since they were the values meant to be predicted.



year	genre1	genre2	genre3	actor1	actor2	actor3	actor4	director	runtime
2011	Action	Drama	0	Ryan Gosling	Bryan Cranston	Albert Brooks	Carey Mulligan	Nicolas Winding Refn	100
2014	Adventure	Drama	Sci-fi	Matthew McConaughey	Anne Hathaway	Jessica Chastain	Mackenzie Foy	Christopher Nolan	169
2003	Documentary	Sport	0	Robert August	0	0	0	Dana Brown	87
2013	Horror	Sci-fi	Thriller	Ethan Hawke	Lena Headey	Max Burkholder	Adelaide Kane	James DeMonaco	85

rating
7.8
8.6
7.4
5.7

Figure 2.13. Tables X and Y of the movie ratings dataset

2.2 HYPERSPECTRAL IMAGING

The dataset collected by the AUM-Hyperspectral team (Kursun et al. 2022), which is yet to be finalized, currently has 462 bands with the goal of cloud detection, classification, and segmentation. The images collected are recorded with the Resonon Pica XC2 camera, which imaging system acts as a “push broom” scanning spectrometer with 462 narrow wavelength bands ranging from 400 nm to 1000 nm. Moreover, regarding the data classification, it is not the shape of the cloud chunks that is the most predictive; in fact, it is the spectral signature due to the scatter/reflective properties of the cloud particles that will help the most for this main task of individual pixel classification (see Figure 2.14). Once the single pixel classification is done effectively, then some form of postprocessing can be performed for segmenting a hyperspectral image into regions according to cloud type or clear sky.

Figure 2.14. A render of a hyperspectral image collected by the AUM-Hyperspectral team.

As it was mentioned before, AUM dataset has not been completed yet, which is why this work will be using two benchmark datasets to test the categorical-boosting-classification method that was developed using CatBoost (Section 3.2). These two datasets are well-known HSI datasets captured by the AVIRIS (Airborne Visible Infrared Imaging Spectrometer) sensor. The first dataset, called Indian Pines, is composed of images of 145×145 pixels in size, with each pixel of the image represented with 204 spectral channels (bands) in the 400-2500 nm range of wavelengths (Grana et al. 2018). The dataset includes 17 classes (class-0 is unlabeled and the other 16 classes are various crops, grass, and woods); Table 2.2 lists the class names and the number of pixels per class in the dataset.

Class ID	Class Name	Number of Pixels
1	Alfalfa	54
2	Corn-notill	1434
3	Corn-mintill	834
4	Corn	234
5	Grass-pasture	497
6	Grass-trees	747
7	Grass-pasture-mowed	26
8	Hay-windrowed	489
9	Oats	20
10	Soybean-notill	968
11	Soybean-mintill	2468
12	Soybean-clean	614
13	Wheat	212
14	Woods	1294
15	Build.-Grass-Trees-Drv.	380
16	Stone-Steel-Towers	95

Table 2.2. The class names and their respective number of pixels of the Indian Pines dataset

With these class labels it is possible to obtain the ground truth image of the HSI picture, which in other words, is the image with the true values that the machine learning will use for identifying the class labels. Figure 2.14 demonstrates the RGB render of the image, while Figure 2.15 shows the class labels (ground truth).

Figure 2.15. An RGB render of the Indian Pines HSI image

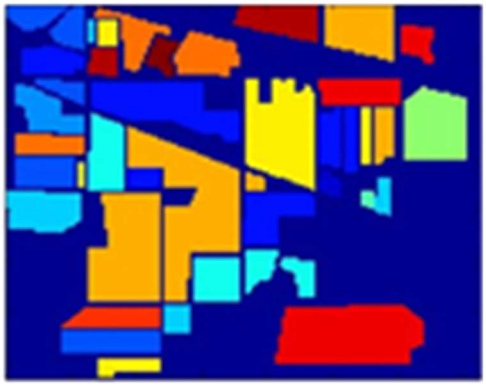


Figure 2.16. Ground truth image of the Indian Pines HSI image

The second dataset, called Salinas, consists of images of 512 × 217 pixels in size with 204 spectral bands (Grana et al. 2018). The class labels and the number of pixels per class in the dataset are listed in Table 2.3.

Class ID	Class Name	Number of Pixels
1	Broccoli (green weeds 1)	2009
2	Broccoli (green weeds 2)	3726
3	Fallow	1976
4	Fallow (rough plow)	1394
5	Fallow (smooth)	2678
6	Stubble	3959
7	Celery	3579
8	Grapes (untrained)	11271
9	Soil (vineyard develop)	6203
10	Corn (senesced green weeds)	3278
11	Lettuce (romaine 4wk)	1068
12	Lettuce (romaine 5wk)	1927
13	Lettuce (romaine 6wk)	916
14	Lettuce (romaine 7wk)	1070
15	Vineyard (untrained)	7268
16	Vineyard (vertical trellis)	1807

Table 2.3. The class names and the number of pixels in each class of the Salinas dataset

These class labels include vegetables and various types of soils where there can be differences in the reflectance (fraction of sunlight reflected from the surface) for the same wavelength interval of each pixel. To demonstrate this, a plot with 5 different class labels was created using their mean wavelength channel value compared to the reflectance, shown in Figure 2.17.

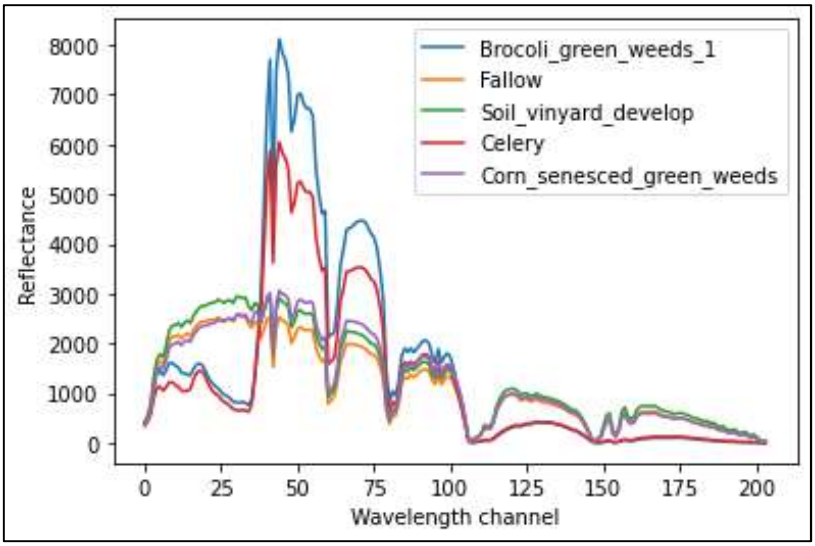


Figure 2.17. Wavelength channel and reflectance values of 5 class labels from the Salinas dataset.

It is also important to clarify that pixels for each of these class labels are similar but not identical. In some cases, with considerable differences between their reflectance values. To demonstrate this, Figure 2.18 shows the wavelength channel and reflectance values for 5 different pixels of the Fallow class-label.

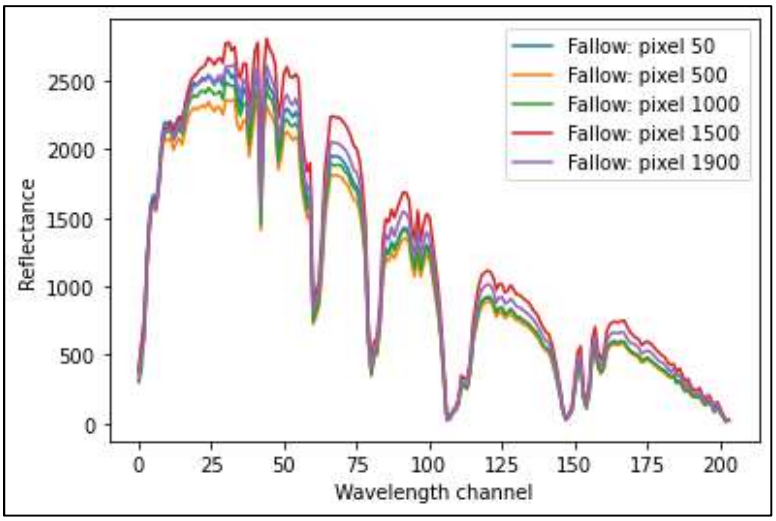


Figure 2.18. Wavelength channel and reflectance values of 5 different pixels for the Fallow class-label

Moreover, the RGB render of the image is shown in Figure 2.19, while the class labels (ground truth) are shown in Figure 2.20.

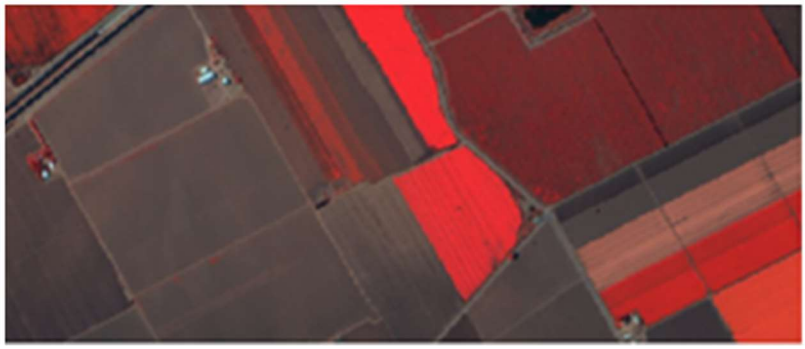


Figure 2.19. An RGB render of the Salinas HSI image

Figure 2.20. Ground truth image of the Indian Pines HSI image

CHAPTER 3: MACHINE LEARNING MODEL

3.1 MOVIE RATINGS

With the use of the Movie Ratings dataset, several approaches were performed utilizing CatBoost in order to test its effectiveness and maximize the chances of a high-accuracy machine learning model (Wang 2022). First, CatBoost was applied directly (Section 3.1.1) without preprocessing data, where its performance was compared to one-hot encoding (Harris & Harris 2012). Then, another approach called the Mean-based approach (Section 3.1.2), focused on calculating the mean of every actor's previous ratings in order to predict future ratings based on those preceding movies. Finally, the last method called the Word-embedding approach (Section 3.1.3), which consisted of applying word embedding to both the actors and genres columns in order to vectorize the data for numerical representation (Mikolov et al. 2013).

3.1.1 DIRECT-CATBOOST APPROACH

This approach was based on directly providing the training and testing data to the CatBoost algorithm with any sort of preprocessing or data conversion. What made this possible was the use of a parameter in the CatBoostRegressor function called "cat_features", which describes the categorical features that will be given to the model. Therefore, all the columns that were considered categorical were inserted on this parameter, which were: actor1, actor2, actor3, actor4, director_name, genre1, genre2, and genre3. Typically, a machine learning model is not able to understand data that is not in numerical format, which is why in situations like this one the common approach is to

apply one-hot encoding. The difference is that with the Catboost algorithm one can automatically do the categorical-to-numerical conversion by specifying which are the categorical features in the model. Therefore, to test the reliability of this automatic conversion, it was decided to apply one-hot encoding before giving the data to the CatboostRegressor function in order to test if the results would be similar (McKinney 2011). This process is exhibited on Table 3.1, where the table only contains genres in order to simplify the demonstration of one-hot encoding.

year	genre1	genre2	genre3	runtime
2004	Action	Adventure	Animation	115
2003	Action	Crime	Thriller	111
2006	Action	Adventure	Thriller	126
2003	Comedy	0	0	85
2002	Drama	Fantasy	History	99

Table 3.1. Table of genres ready for one-hot encoding

Then, for every different categorical value present in this table, a new column is created that will contain a 1 or 0 depending on whether the rating of a specific row has that category or not. A demonstration of this is displayed on Table 3.2.

year	runtime	Action	Adventure	Animation	Crime	Thriller	Comedy	Drama	Fantasy	History
2004	115	1	1	1	0	0	0	0	0	0
2003	111	1	0	0	1	1	0	0	0	0
2006	126	1	1	0	0	1	0	0	0	0
2003	85	0	0	0	0	0	1	0	0	0
2002	99	0	0	0	0	0	0	1	1	1

Table 3.2. Table of genres after one-hot encoding

It is important to point out that a variation of one-hot encoding was used in this thesis. Instead of inserting a value of 1 for every row that had a present genre in one of the columns, it was decided to use values of 1, 0.8 or 0.6 depending on the situation. If a movie had its principal genre (genre1) in the column, then it would receive a 1. If it contained a secondary genre (genre2) then it would be a 0.8. If it had a third genre (genre3) then a 0.6. The logic behind this is that the most relevant genre of the movie should receive a heavier weight in the calculation of the resultant rating. The machine learning model was able to understand this by using the aforementioned variation of one-hot encoding.

Then, after running both the one-hot encoding and Direct-Catboost programs using genres as the training data, the one-hot encoding version had approximately the same results of the Direct-Catboost version, obtaining 0.60 and 0.58 respectively, proving this way that Catboost can easily convert categorical data into numerical with only the use of an extra parameter in its function and still produce similar results as manually converting that data.

With this knowledge in hand, there was enough confidence to give the CatboostRegressor function all the categorical data at once (including actors and directors). This extra data provided, was able to improve the model performance to a coefficient of correlation of 0.65. In a nutshell, Figure 3.1 showcases the algorithm for the Direct-CatBoost approach.

```
DIRECT-CATBOOST ALGORITHM

1)Select columns: year, genre1, genre2, genre3, actor1,
actor2, actor3, actor4, director_name, runtime for table X
and ratings for table Y

2) Train-test split the data using 0.15 as the size of the
split parameter.

3) Declare the CatBoostRegressor function using the
categorical data (genre1, genre2, genre3, actor1, actor2,
actor3, actor4, director_name) in cat_features

4) Fit the model using X_train, Y_train, and then predict
the X_test table

5) Compare y_test with y_pred in order to calculate the R-
score correlation result
```

Figure 3.1. Pseudocode for the Direct-Catboost approach program

3.1.2 MEAN-BASED APPROACH

This model consisted on selecting every distinct person of an actor or director column and calculating a mean rating for them. In order to implement this, a Python dictionary to store the mean rating of every different item was needed, where the key of the dictionary was the actor or director in the column, and the respective value was the mean rating of that person. For every movie in the dataset, the four actor columns and the director column were selected, along with their ratings, year of release, and runtime. The reason for only selecting these columns in this approach, was to determine if previous successes or failures of actors and directors could be a factor of determining future ratings for upcoming movies. The concept behind this approach is explained in Eq. 1.

$$X = \frac{\sum_{k=1}^{n} X_k}{n} \quad (1)$$

Where:

X_k = the rating of the kth movie

n = number of movies

X = the mean rating

The mean rating X was calculated according to Eq. 1 by summing together each of the person's previous movie ratings ($X_1, X_2, X_3 \ldots X_k$) and dividing by the number of movies that person had previously acted or directed on. A demonstration of how the data was transformed is given in Tables 3.3, 3.4, and 3.5.

year	actor1	actor2	actor3	actor4	genre1	genre2	genre3	director	runtime
2003	Craig T. Nelson	Samuel L. Jackson	Holly Hunter	Jason Lee	Action	Adventure	Animation	Brad Bird	115
2003	Donald Sutherland	Mark Wahlberg	Edward Norton	Charlize Theron	Action	Crime	Thriller	F. Gary Gray	111
2006	Tom Cruise	Michelle Monaghan	Ving Rhames	Philip Seymour Hoffman	Action	Adventure	Thriller	J.J. Abrams	126
2003	Adam Goldberg	Andy Dick	Judy Greer	Mario Van Peebles	Comedy	0	0	Jonathan Kesselman	85
2002	Sergey Dreyden	Mariya Kuznetsova	Leonid Mozgovoy	0	Drama	Fantasy	History	Aleksandr Sokurov	99

Table 3.3 Initial table of values for the Mean-based approach

Then, the genres are removed since they will not be considered on this approach (mean values will only be calculated for actors and directors). This is shown in Table 3.4.

year	actor1	actor2	actor3	actor4	director	runtime
2003	Craig T. Nelson	Samuel L. Jackson	Holly Hunter	Jason Lee	Brad Bird	115
2003	Donald Sutherland	Mark Wahlberg	Edward Norton	Charlize Theron	F. Gary Gray	111
2006	Tom Cruise	Michelle Monaghan	Ving Rhames	Philip Seymour Hoffman	J.J. Abrams	126
2003	Adam Goldberg	Andy Dick	Judy Greer	Mario Van Peebles	Jonathan Kesselman	85
2002	Sergey Dreyden	Mariya Kuznetsova	Leonid Mozgovoy	0	Aleksandr Sokurov	99

Table 3.4 Table containing only actors and directors before transforming values

Finally, the mean values are calculated for every categorical data in the table, demonstrated in Table 3.5.

year	actor1	actor2	actor3	actor4	director	runtime
2003	8.0	6.32	6.68	6.15	7.46	115
2003	6.88	6.47	7.14	6.75	6.76	111
2006	6.86	6.39	6.58	6.56	7.22	126
2003	6.1	5.85	6.43	6.1	5.65	85
2002	7.4	7.4	7.4	7.4	7.4	99

Table 3.5 Table containing only actors and directors after transforming values

As shown in Table 3.5, every actor and director received a score based on their previous movies and their respective ratings. There were rare scenarios when one or many members of the cast had not been in a movie ever before, these cases were handled by giving those persons an average rating based on the rest of the cast ratings. An example of this is shown in the last row of Table 3.5 where only one actor had previous movies in the dataset. Therefore, the mean rating of that actor was used for rest of the

cast. Likewise, there can also be even a rarer scenario which is the case when a movie has an entire cast that has never acted/directed in a film before. In this case, there was no choice but to give a value of 6 to every member of the cast (a common average rating) since there was no possible way to assign a mean rating for any member of the cast in that movie. In conclusion, the final algorithm for the Mean-based approached is demonstrated in Figure 3.2.

MEAN-BASED ALGORITHM

```
1) Select columns: year, actor1, actor2, actor3, actor4,
director_name, runtime for table X and ratings for table Y

2) Train-test split the data using 0.15 as the size of the
split parameter.

3) Store every distinct item of the actor1, actor2, actor3,
actor4, and director_name columns in a dictionary, and use
each of these items (the persons' names) as their key.
Then, for every key, their value will be the mean rating of
that person.

4) Transform the items of X_train that are present in the
dictionary from the keys to their corresponding values

5) Repeat step 4 for X_test but create an exception in the
scenario where the key is not found in the dictionary
(value of 0). To handle the 0s, calculate an average of the
items of the same row as the 0 is, and replace every
instance of that 0 with the calculated average number.

5) Fit the model using X_train,Y_train, and then predict
the X_test table

6) Compare y_test with y_pred in order to calculate the R-
score correlation result
```

Figure 3.2. Pseudocode for the Mean-based approach program

3.1.3 WORD EMBEDDING APPROACH

This model was based on representing words (string values) for text analysis, where every string received by the model was transformed into numerical vector representations. The idea was that if words had similar meaning, they would be represented with similar vector values (Goldberg et al. 2014). Every categorical feature in the dataset was stored in a list where the vectorization took place (Rehurek & Sojka 2011). For every row of the dataset there would be two different vector representations taking place: one for the genres of the movie and another for the actors and director of that title (Oliphant 2006).

Specifically, every row would convert their genre values into a 40-dimensional vector representation, and the movie's cast to another 40-dimensional representation. One thing to note is that in this thesis, for each of the forty dimensions, the first twenty were always assigned for only the first actor or first genre, and then the other twenty dimensions would be used to represent the rest of the cast or the rest of the genres. This was done in order to test how much of an impact the lead actor and lead genre could make in the ratings prediction. After this, a new table was created which contained all these new vector values, so that the machine learning model could apply the training and testing necessary. Tables 3.7 and 3.8 demonstrate the data before and after the vectorization took place.

year	actor1	actor2	actor3	actor4	genre1	genre2	genre3	director	runtime
1980	Anthony Hopkins	John Hurt	Anne Bancroft	John Gielgud	Biography	Drama	0	David Lynch	124
2008	Ray Stevenson	Dominic West	Julie Benz	Doug Hutchison	Action	Crime	Drama	Lexi Alexander	103
2014	Cameron Diaz	Leslie Mann	Nikolaj Coster-Waldau	Kate Upton	Comedy	Romance	0	Nick Cassavetes	109
2010	Benicio Del Toro	Anthony Hopkins	Simon Merrells	Emily Blunt	Drama	Fantasy	Horror	Joe Johnston	103
1999	Chris O'Donnell	Renée Zellweger	Artie Lange	Ed Asner	Comedy	Romance	0	Gary Sinyor	101

Table 3.7. Initial table of values for the Word-embedding approach

year	runtime	gd1	gd2	gd3	…	gd38	gd39	gd40	pd1	pd2	pd3	…	pd38	pd39	pd40
1980	124	-2.235	0.092	2.895	…	-1.219	-2.284	-1.322	-0.008	-0.034	-0.003	…	-0.025	-0.009	0.006
2008	103	-2.235	0.092	2.895	…	-0.822	-1.868	-1.143	0.009	-0.023	0.035	…	-0.066	-0.074	-0.036
2014	109	-1.282	-0.128	2.234	…	-1.412	-3.251	-2.057	-0.015	0.033	-0.041	…	-0.014	-0.023	-0.014
2010	103	-0.553	0.028	0.657	…	-0.349	-0.705	-0.486	0.049	0.045	0.014	…	0.016	-0.023	-0.013
1999	101	-2.235	0.092	2.895	…	-0.952	-2.433	-1.427	0.010	-0.013	-0.009	…	-0.033	-0.038	-0.0007

Table 3.8. Table of values after word-embedding vectorization

As shown on Table 3.8, the table consisted of 82 columns where every gd column stood for “genre dimension” and every pd column for “person dimension”. Several dimension sizes were tried (from 10 to 80), but 40 was the best option in terms of results. Lastly, in Figure 3.3 the algorithm for the Word-embedding approach is shown.

WORD-EMBEDDING ALGORITHM

```
1)Select columns: year, genre1, genre2, genre3, actor1, actor2,
actor3, actor4, director_name, runtime for table X and ratings
for table Y

2) Train-test split the data using 0.15 as the size of the split
parameter.

3) Store every column that contains categorical data from X_train
into a list including their missing values (0's)

4) Copy the values of the list (without the 0s) to another list.
Then, apply Word2Vec() model with vector_size = 20 to this new
list

5) Create a for loop to assign every vector value of the genre1
column into a list, then do the same for actor1, but store their
vector values into a second list.

6) Create a for loop to assign every vector value of genre 2 and
genre 3 into a third list, and then actor 2, actor 3, actor 4,
director_name into a fourth list.

7) Use the first list that has the vectors of genre 1 to create
20 dimensions (20 columns). Then, use the second list which has
the vectors of actor1 to create 20 more dimensions (20 columns).

8) Repeat step 7 but using the third list (containing genre2,
genre3) to create 20 dimensions. Then use the fourth list
(containing actor2, actor3, actor4, director_name) to create 20
more dimensions.

9) Append the dimensions of genres and actors respectively to
have a total of 40 columns (dimensions) of genres vector values,
and 40 columns (dimensions) of people (actors and directors),
then append this 80-dimensional table to X_train

10) Do steps 3 through 9 but for X_test instead

11) Fit the model using X_train,Y_train, and then predict the
X_test table

12) Compare y_test with y_pred to calculate the R-score
```

Figure 3.3. Pseudocode for the Word-embedding approach program

3.1.4 MODEL ROADMAP

This section depicts the whole process, from start to finish, of all the necessary steps needed in order to calculate the movie ratings predictions using any of the approaches explained in this section. Figure 3.4 demonstrates a roadmap with each of these important steps (Taei 2022).

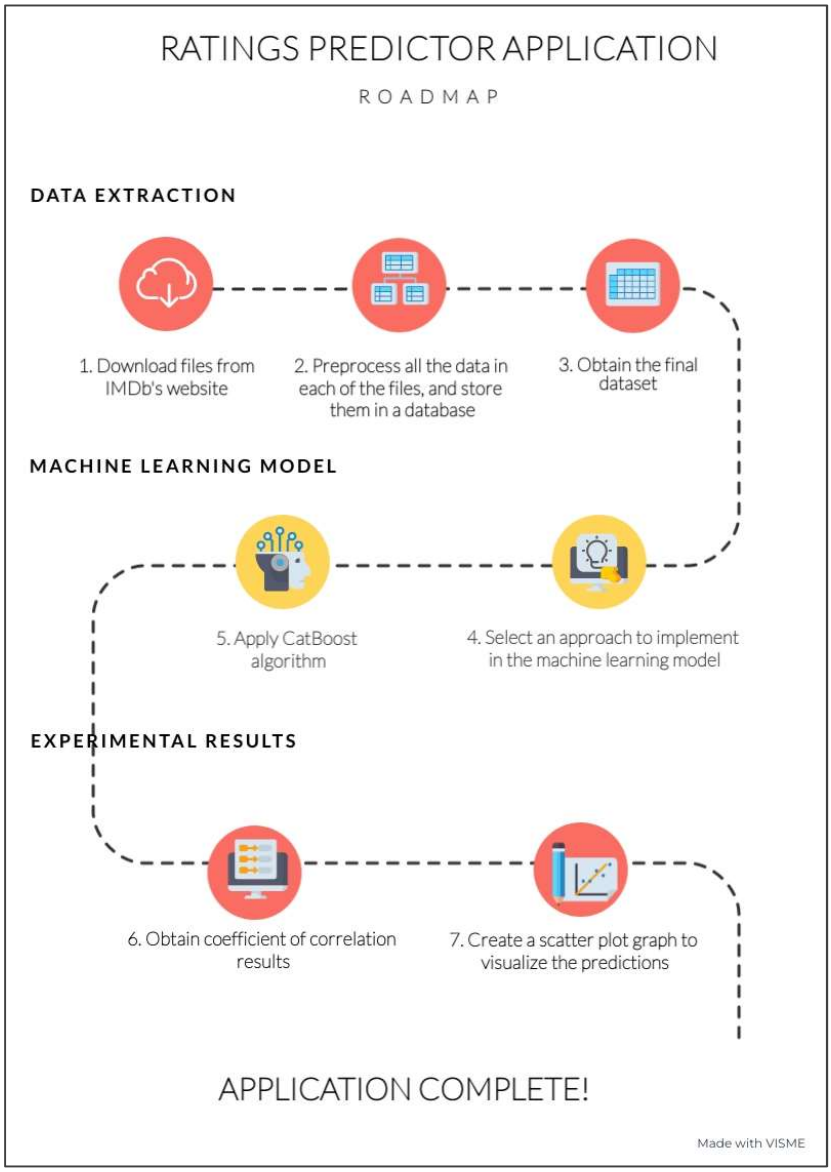


Figure 3.4. Ratings Predictor Application roadmap

3.2 HYPERSPECTRAL IMAGING

While the high-resolution representation of an individual pixel in HSI (having many narrow wavelengths covering a large portion of the spectrum from near-ultraviolet to near-infrared range) makes discrimination of many more classes from each other using just a single pixel (GISGeography 2022), reflectivity/irradiances in nearby wavelength intervals are generally very redundant and dimensionality reduction methods (Alpaydin 2014) are needed for band selection for HSI systems (Sellami et al. 2019).

Applying some feature selection algorithms with high time complexity such as sequential backward selection (with time complexity of $O(n^2)$) is almost prohibitive for such high dimensional datasets. Moreover, feature selection algorithms need to be further adapted to HSI domain, because the band selection for the classification task should also help determine important ranges of the spectrum. Feature selection process should not necessarily treat each one of the hundreds of wavelengths of the spectrum as separate or unrelated variables, because selection of individual wavelengths of the spectrum may not be justified and the task could be simplified by finding a few wavelength ranges of greatest importance.

For example, the AUM-hyperspectral team has identified four intervals to be the most useful for cloud classification. As shown in Figure 3.5, there are four intervals (highlighted with red dash-lines) identified by the domain expert Randy Russell in the team. Representing one interval as one categorical variable (i.e., creating a categorical variable for that interval by representing it with the index of clustering applied to the dataset using the variables in that interval only) can be a good approach to utilize and test

the capabilities and applicability of CatBoost or other comparable boosting methods to the HSI domain.

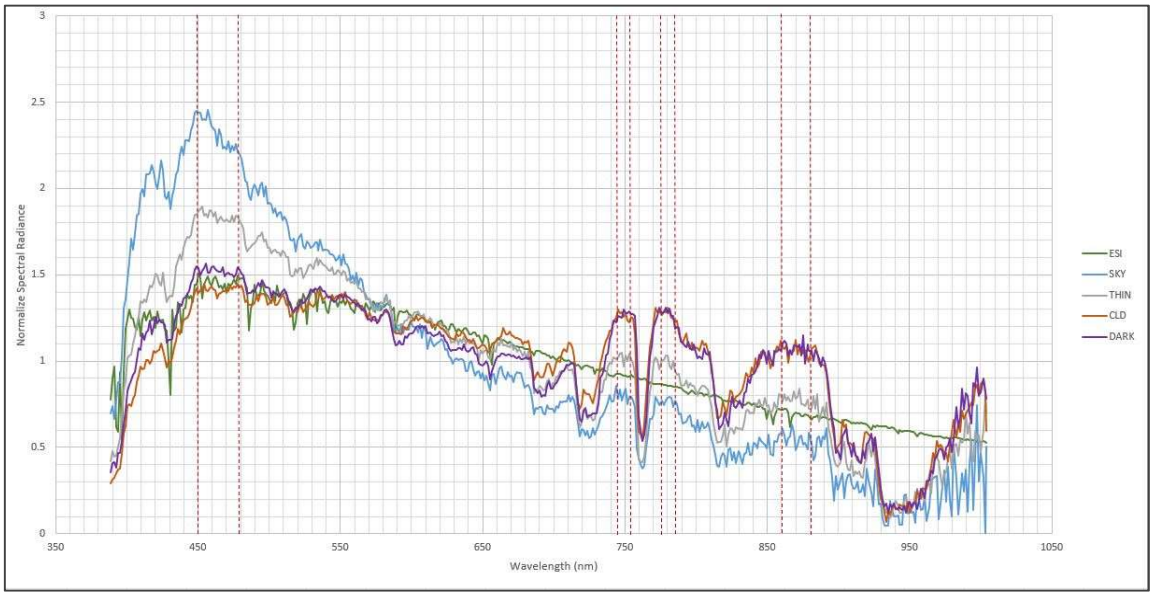


Figure 3.5. Signatures of clear sky (SKY), thin clouds (THIN), clouds (CLD), and dark clouds (DARK) in 462 bands of the HSI images being collected by the AUM Hyperspectral team.

The proposed algorithm called Clustered-Shifting-Window Boosting Algorithm presented in Figure 3.6 is first applied on the two benchmark datasets: Indian Pines and Salinas. The algorithm will be tested on the AUM HSI Cloud dataset once it is finalized.

```
Clustered-Shifting-Window Boosting Algorithm
Inputs:
     X[N,D]: Train-set of N samples and D features
     y[N]: Class-labels of the N samples

     X_test[M,D]: Test-set of M samples and D features
     Y_test[M]: Class-labels of the M test samples

     w: Window length
     s: Stride of windows
     K: Number of clusters in each window

Output:
     Model: Classifier (and centroids)
     Acc: Accuracy on the test set

Begin:

Num_Windows = 0

for win_start = 1:s:D
     win_end = win_start + w
     windowed_data = X[:, win_start:win_end]
     centers = Kmeans(windowed_data, K)

     train_centers = Find_Nearest_Center(X, centers)
     test_centers = Find_Nearest_Center(X_test, centers)

     Categorical_Trainset[:, Num_Windows] = train_centers
     Categorical_Testset[:, Num_Windows] = test_centers
     Num_Windows = Num_Windows + 1

end for

Model = Train_Boost_Classifier(Categorical_Trainset, y)
Acc = Test_Classifier(Model, Categorical_Testset, y_test)
```

Figure 3.6. Clustered-Shifting-Window Boosting Algorithm for HSI dataset

3.2.1 MODEL ROADMAP

This section demonstrates all the necessary steps needed in order to predict pixels from an HSI image. Figure 3.7 demonstrates a roadmap with each of these important steps (Taei 2022).

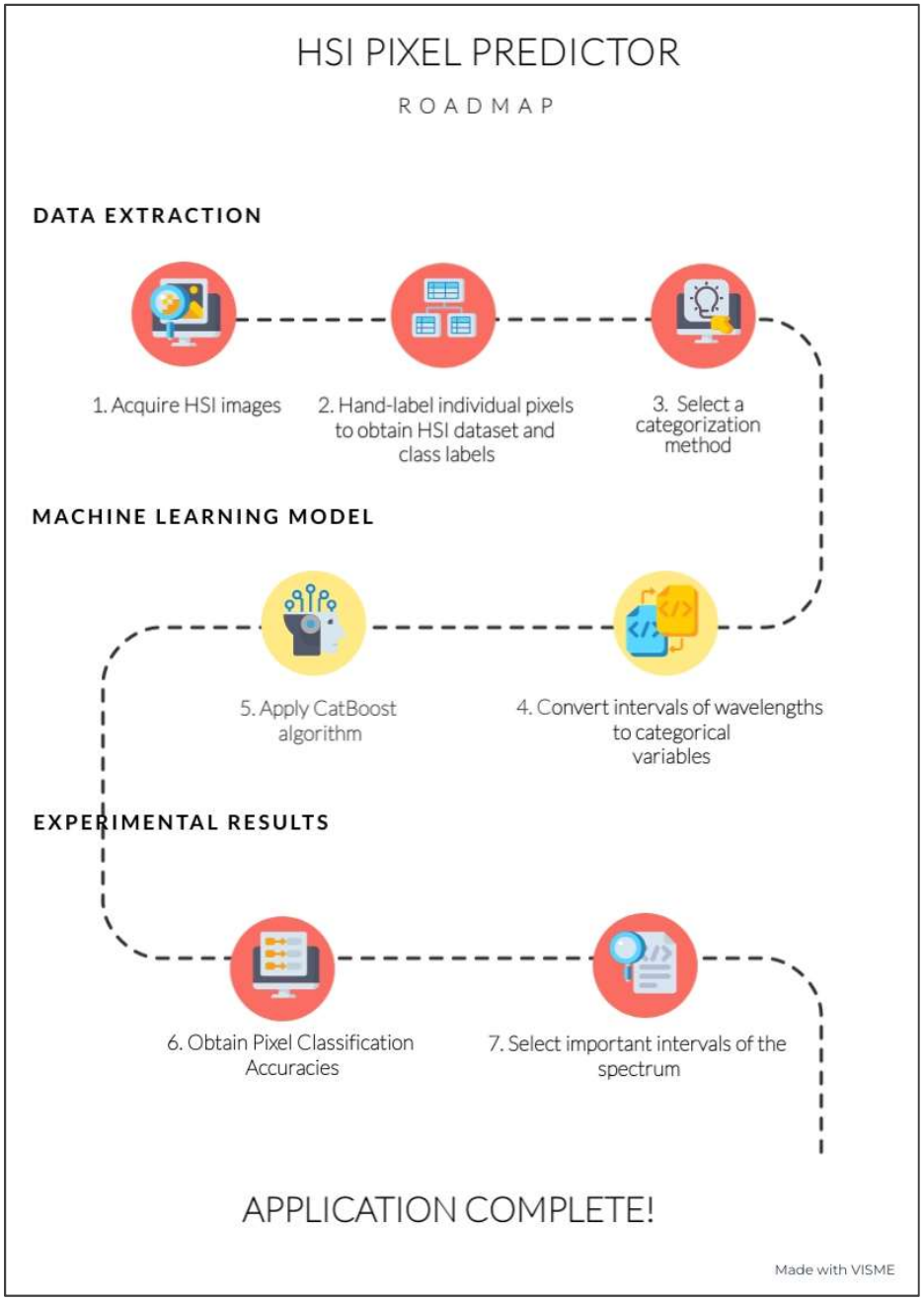


Figure 3.7. HSI Pixel Predictor Roadmap

CHAPTER 4: EXPERIMENTAL RESULTS

In this chapter, several results were calculated from the Movie Ratings dataset (Section 4.1) utilizing the approaches previously explained (Section 3.1. Regarding hyperspectral imaging, results of the HSI classification (Section 4.2) were calculated with their respective sensitivity of different hyperparameters.

4.1 MOVIE RATINGS RESULTS

For each of the approaches in this section, their results are demonstrated calculating the coefficient correlation of the model and displaying a scatterplot to visualize the predictions results.

4.1.1 DIRECT-CATBOOST APPROACH

This model obtained 0.65 coefficient of correlation, which was the highest accuracy achieved on this thesis for the Movie Ratings dataset. The result can be showcased in the scatterplot in Figure 4.1.

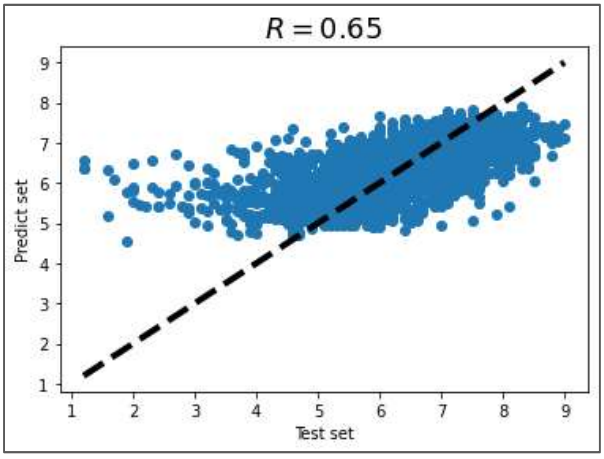


Figure 4.1. Scatterplot result for the Direct-Catboost approach

4.1.2 MEAN-BASED APPROACH

This model obtained 0.58 coefficient of correlation, which was promising since it only used actors and directors without the use of genres columns. This proved that these columns are essential for predicting unseen data on this dataset. The scatterplot in Figure 4.2 displays this result.

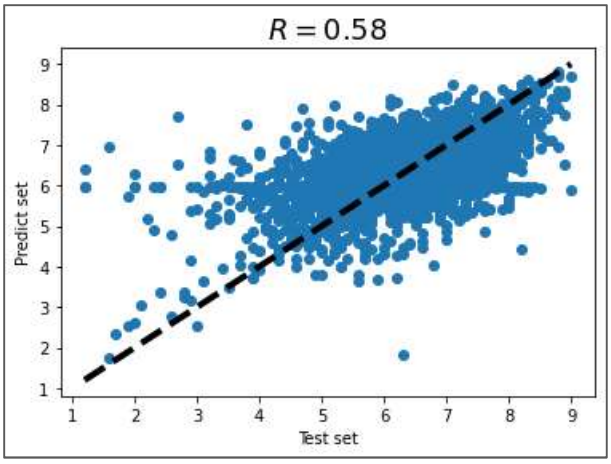


Figure 4.2. Scatterplot result for the Mean-based approach

4.1.3 WORD-EMBEDDING APPROACH

This model obtained 0.45 coefficient of correlation, which was the lowest score from all of the approaches. It can be demonstrated that word embedding was not remarkably successful for this dataset, especially because it could not predict values with low ratings as shown in Figure 4.3.

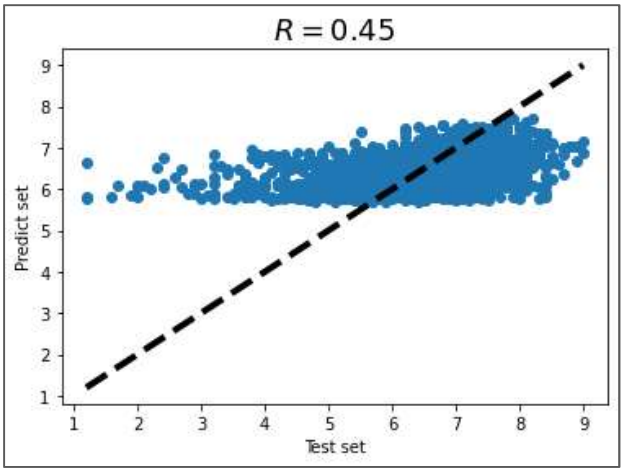


Figure 4.3. Scatterplot result for the Word-embedding approach

4.2 HYPERSPECTRAL IMAGING RESULTS

As shown in Table 4.1 (using the Salinas dataset), for each window, a single categorical variable with K categories is created by K-means and these are stacked horizontally to create a dataset with *clustered categorical features* to train/test the boosting model. The window length, *w*, and the stride amount, *s*, determine the number of categorical features created (i.e. the number of intervals). The results reported are the means and standard deviations of 10 runs (each with 10% vs 90% train-test splits).

	Stride = 5 Window length = 10	**Stride = 10 Window length = 10**	**Stride = 5 Window length = 20**	**Stride = 10 Window length = 20**
K = 10	86.93 ± 0.32	85.39 ± 0.21	86.74 ± 0.48	86.06 ± 0.58
K = 20	88.06 ± 0.24	86.70 ± 0.16	88.73 ± 0.17	87.89 ± 0.28
K = 30	88.36 ± 0.13	86.95 ± 0.15	89.13 ± 0.13	88.5 ± 0.19

Table 4.1. Sensitivity of the hyperparameters (various K for K-means, window length and window stride) for Salinas dataset.

As the categorical dataset has fewer features, it is more suitable for feature selection and classification using boosting, which is among the goals in this line of work. In the pixel classification task of 17 classes (16 target classes of various vegetation etc. and the unlabeled pixels with class-label of 0) on the test set, accuracies with the proposed boosting method yields favorable results. For comparison with the use of the original raw variables (i.e., 204 bands), various popular benchmark classifiers (Fernandez-Delgado et al. 2014; Pedregosa et al. 2011) are also used. The results on the Indian Pines and the Salinas datasets are reported in Table 4.2. As before in Table 4.1, the results reported in Table 4.2 are the means and standard deviations of 10 runs (each with 10% vs 90% train-test splits).

	Indian Pines	**Salinas**
K-NN (K=1)	63.49 ± 0.28	86.61 ± 0.11
K-NN (K=3)	65.87 ± 0.40	87.30 ± 0.14
K-NN (K=5)	66.81 ± 0.32	87.40 ± 0.14
Boosting on the original features	71.88 ± 0.55	88.38 ± 0.12
Proposed K-means + Boosting	71.89 ± 0.60	89.13 ± 0.13

Table 4.2. Comparisons on HSI classification results

CHAPTER 5: CONCLUSION

In this thesis, machine learning algorithms based on categorical boosting were applied to different datasets with the goal of obtaining high-accuracy results with dimensionality reduction. In the Movie Ratings dataset, CatBoost was used to handle categorical data effectively by directly specifying the categories on the CatBoost regressor function (instead of trying the infeasible one-hot-encoding or trying to adapt a word-embedding method to deal with the genre and actor categories in the data). Efforts in adapting/applying the boosting algorithms studied in this thesis to the hyperspectral cloud segmentation project conducted by the AUM Hyperspectral team led to an algorithm that was named the Clustered-Shifting-Window Boosting Algorithm for classifying hyperspectral image pixels. The proposed algorithm applies K-Means clustering to create new categorical variables, which can be used either to reduce dimensionality or to augment the original HSI dataset. Experiments on the Indian Pines and Salinas datasets showed favorable results.

As future work, the algorithms developed in this thesis will be applied to the final version of the cloud dataset that is currently under preparation for the AUM-Hyperspectral research grant.

REFERENCES

Alpaydin, E. (2014). *Introduction to machine learning, third edition*. The MIT Press, Cambridge.

Fernandez-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181.

GISGeography (2022) Multispectral vs Hyperspectral Imagery Explained. https://gisgeography.com/multispectral-vs-hyperspectral-imagery-explained/. Last accessed on 22 of July of 2022.

Goldberg, Y. & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, *arXiv.*

Grana, M., Veganzons, M., & Ayerdi, B. (2018). Hyperspectral remote sensing scenes - grupo de inteligencia computacional (GIC). http://www.ehu.eus/ccwintco/index.php. (Accessed on 12/22/2018).

Gulin, A. et al. (2018). CatBoost: unbiased boosting with categorical features, *arXiv.*

Harris, D. & Harris, S. (2012). *Digital design and computer architecture (2nd ed.)*. San Francisco, Calif.: Morgan Kaufmann. p. 129.

Hunter, J. (2007). Matplotlib: A 2D graphics environment, Computing in Science & Engineering

IMDb datasets. IMDb, https://www.imdb.com/interfaces/ . Accessed 20 December 2021

Kursun, O., Cueva-Parra, L., & Russell, R. (2022). NSF RUI: Collaborative Research: CDS&E: A Modular Multilayer Framework for Real-Time Hyperspectral Image Segmentation National Science Foundation Grant No. 2003740.

Kursun, O., Dinc, S., & Favorov, O.V. (2021). Contextually Guided Convolutional Neural Networks for Learning Most Transferable Representations. ArXiv:2103.01566 [Cs], Mar. 2021. arXiv.org, http://arxiv.org/abs/2103.01566

McKinney, W. (2011). pandas: A Foundational Python Library for Data Analysis and Statistics

Mikolov, T. et al. (2013). Distributed Representations of Words and Phrases and their Compositionality, *arXiv.*

Oliphant, T. (2006). Guide to NumPy

Oracle MySQLWorkbench (Version 8.0) https://dev.mysql.com/downloads/workbench/ . Accessed 23 December 2021

Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python*, JMLR,* 12, pp. 2825-2830.

Rehurek, A. & Sojka P. (2011). Gensim–python framework for vector space modelling, NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic

Samat et al. (2021) GPU-Accelerated CatBoost-Forest for Hyperspectral Image Classification Via Parallelized mRMR Ensemble Subspace Feature Selection*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14: 3200 - 3214. https://ieeexplore.ieee.org/document/9368975

Sellami, A., Farah, M., Riadh Farah, I., & Solaiman, B. (2019). Hyperspectral imagery classification based on semisupervised 3-d deep neural network and adaptive band selection. *Expert Systems with Applications*, 129:246 – 259.

Taei, P. (2022). Visme (version 3.4.2) https://www.visme.co/es/.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Wang, P. (2022). Anaconda (version 2.2.0) https://anaconda.org/anaconda/spyder.