Patter Recognition – Phase 2
Classification

Cover Letter

Team_ID: CS_56
Project: Movie

#	*NAME*	*SEAT NUMBERS*	*DEPARTMENT*
1.	عمر احمد فاروق احمد الجبالى	20201701162	CS
2.	عماد محمود عمر الفاروق محمود	20201700517	CS
3.	عمر احمد محمد عبدالمعطى	20201700524	CS
4.	عمر عبدالفتاح عبد السميع ابراهيم الخولي	20201700538	CS
5.	محمد ابراهيم سعيد امام	20201700650	CS
6.	محمد اسامه كمال يوسف محمد	20201700665	CS

Table of Contents

Preprocessing

Some features have the same appropriate format to deal with like in regression phase and some not, so some features handled in suitable way to convert these features to more ready features.

Here are Features Preprocessing Techniques that will change in Classificaiton:

List Of Dictionaries Features

There were 7 features consists of many of values in each row and each value has the same structure of details as follows:

Feature	***Selected-Key***
genres	id
keyword	id
production_companies	id
production_countries	name
spoken_languages	iso_639_1
cast	name
crew	name

- **The approach used in regression phase *(weighted average of conentration points)* will fail:**
  - When extracting all possible values of specific detail in each feature, its not reslting a one binary array, but it results number of binary arrays equals to number of labels in target.
  - When trying to find concentration points and calculate the average weight by this method, you discard some important information like how row is sure of High, Intermediate, and Low.

- **Used approach can be defined as follows:**
  - Extract all possible values of specific detail in each feature, then sort these values on its frequency in current label so that the most left values (close to 0 index) have lower frequency in label and the most right values (close to 1 index) have higher frequency in label.
  - On each feature apply, extract all values in current row of that feature then calculate summation of the indcies of each occurence of value in a label.

- Extract 3 Features called ‘FeatureName’_High, ‘FeatureName’_Intermediate, and ‘FeatureName’_Low and drop the original feature.

- The new 3 features now store how row is more confidence of that label by the index **(rigth/left most).**

Feature Extraction on Budget/Revenue

Instead of using Pearson Correlation that doesn't work in categorical targets, in this phase using ANOVA was better for the case as follows:

# (Best to Worst)	Feature	F - Value	P - Value
1	Profitable	88.006926	7.047089e-38
2	Profit	74.434200	2.771213e-32
3	Revenue	58.623864	1.047671e-25
4	Budget	11.616859	9.421214e-06

- **F-Value:** This column shows the ANOVA F-value for each feature, which measures how much variance in the target variable (Rate) is explained by that feature. Features with higher F-values are more strongly associated with the target variable.

- **P-Value:** This column shows the p-value for each feature, which measures the statistical significance of the association between that feature and the target variable. Features with lower p-values are more likely to be truly associated with the target variable, rather than being due to random chance.

Based on this output, we can see that:

- **The "profitable" feature** has the highest F-values and the lowest p-values. This suggests that it is the most strongly associated with the target variable (Rate) and would be good candidates to include in a predictive model.

- **The "profit", and "revenue" features** have a lower F-value and higher p-value, indicating that it may be less important or less statistically significant as a predictor.

- **The "budget" feature** has lowest F-value and the highest p-value, indicating that it is least important or least statistically significant as a predictor.

Feature Selection using ANOVA Algorithm

- Works on Numerical Inputs and Categorical Outputs.
- **F-Value:** This column shows the ANOVA F-value for each feature, which measures how much variance in the target variable (Rate) is explained by that feature. Features with higher F-values are more strongly associated with the target variable.

- **P-Value:** This column shows the p-value for each feature, which measures the statistical significance of the association between that feature and the target variable. Features with lower p-values are more likely to be truly associated with the target variable, rather than being due to random chance.

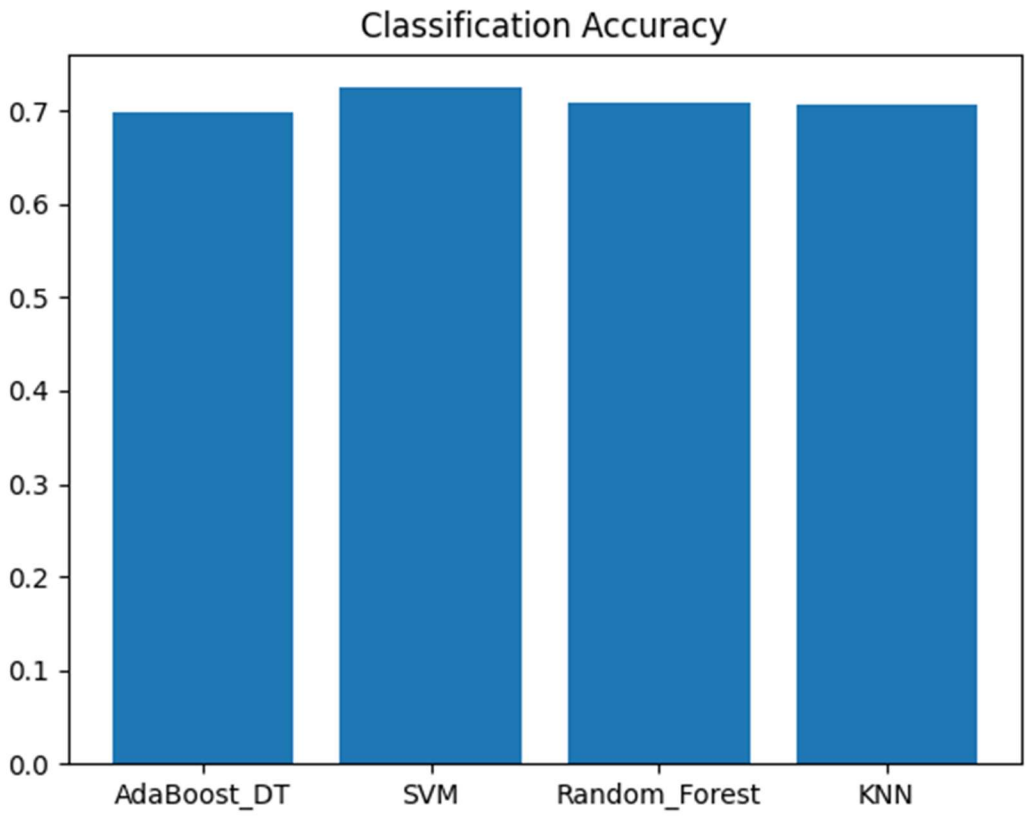
# (Best to Worst)	Feature	F - Value	P - Value
1	cast_high	237.785648	5.11E-95
2	runtime	199.734215	5.09E-81
3	keywords_high	174.442694	1.67E-71
4	vote_count	142.466146	3.19E-59
5	crew_high	128.448193	9.53E-54
6	viewercount	71.443652	7.13E-31
7	profitable	69.920104	3.01E-30
8	production_companies_high	68.979931	7.31E-30
9	cast_intermediate	66.855798	5.47E-29
10	production_companies_intermediate	47.806014	4.34E-21
11	crew_intermediate	38.211769	4.58E-17
12	keywords_intermediate	28.463088	6.05E-13
13	release_date	23.912068	5.20E-11
14	genres_low	18.643489	9.22E-09
15	cast_low	15.936945	1.33E-07
16	keywords_low	13.804205	1.09E-06
17	production_companies_low	13.21894	1.95E-06
18	genres_intermediate	11.303961	1.30E-05
19	genres_high	9.512764	7.67E-05
20	production_countries_high	6.714342	1.24E-03
21	crew_low	6.616122	1.36E-03
22	spoken_languages_high	6.534667	1.48E-03
23	homepage	6.156396	2.15E-03
24	overview	5.550138	3.94E-03
25	status	5.379262	4.67E-03
26	production_countries_intermediate	4.96889	7.02E-03
27	spoken_languages_intermediate	4.920145	7.37E-03
28	tagline	4.103224	1.66E-02
29	spoken_languages_low	3.436921	3.23E-02
30	production_countries_low	3.278703	3.78E-02
31	original_language	3.034671	4.83E-02
32	id	2.664319	6.99E-02
33	title	0.298276	7.42E-01
34	original_title	0.152956	8.58E-01

- Select the features highlighted above that affects the more on target column depends on what each model needs *(See Classification Techniques)*.

Train & Test Split

- Apply Train Test split on the merged datasets:
  - 80% for Train -> **movies_train**
  - 20% for Test -> **movies_test**

Classifcation Techniques



Classification Model #1: Decision Tree Classification

Classification Results

MEASURE	***RESULT***
Accuracy	0.7199341021416804
Precision	0.7003156565656566
Recall	0.5121735562912034
F1-Score	0.5427719821162444

Hyper-Parameter Tuning

TOP 29 Features Selected			
Decision Tree (Estimator)		***AdaBoost (Boosting)***	
Hyper-Parameter	***Value***	***Hyper-Parameter***	***Value***
Class Weight	None	***Algorithm***	'SAMME'
Criterion	'gini'	***N Estimators***	100
Max Depth	10	***Random State***	None
Min Samples Leaf	2	***Learning Rate***	1.0
Min Samples Split	5		

Notes & insights

When trying to use more complex hyper-parameters on adaboosting algorithm like n_estimators, learning_rate, or random_state, it leads to more inaccurate results (bad precision, recall, f1-score) example getting f1-score with 0.29 value but in some cases have some good accuracy values realative to the best one in above table.

Classification Model #2: SVM Classification

Classification Results

MEASURE	***RESULT***
Accuracy	0.728171334431631
Precision	0.8103056090772854
Recall	0.5111243150458836
F1-Score	0.513647046621871

Hyper-Parameter Tuning

TOP 16 Features Selected			
SVC (Estimator)		Bagging Classifier	
Hyper-Parameter	Value	Hyper-Parameter	Value
C	100	***N Estimators***	100
Gamma	0.1		
Kernel	'linear'		

Notes & insights

Without Bagging

1. **One vs. One:**
   - ❖ A model used for multiclass classification SVM, IN (One vs. One) we train SVM model for each classifier pair, when making prediction all classifiers would make a prediction.
     i. high vs low
     ii. high vs intermediate
     iii. intermediate vs low
2. **One vs. All:**
   - ❖ A model used for multiclass classification SVM, In (One vs. All) we train SVM model for each classifier as a positive class and the rest as a negative class, all classifiers make prediction and the class with the highest confidence score would be chosen as the final prediction.
     i. high vs. (low & intermediate)
     ii. low vs. (high & intermediate)
     iii. intermediate vs. (low & high)

With Bagging

1. **One vs. One:**
   - ❖ Applying one vs. all with bagging to improve accuracy, precision, recall and f1-score after applying Tunning paramter on bagging parameter first we choose n_estimator = 100,200,1000,1500 by iterating we found (1000) the best value of n_estimators
     i. high vs low
     ii. high vs intermediate
     iii. intermediate vs low

2. **One vs. All:**
   - ❖ Applying one vs. one with bagging to improve accuracy, precision, recall and f1-score after applying Tunning paramter on bagging parameter first we choose

n_estimator = 100,200,1000,1500 by iterating we found (1000) the best value of n_estimators

i. high vs. (low & intermediate)
ii. low vs. (high & intermediate)
iii. intermediate vs. (low & high)

Classification Model #3: Random Forest Classification

Classification Results

MEASURE	***RESULT***
Accuracy	0.71334431630972
Precision	0.7029569892473119
Recall	0.478675645342312
F1-Score	0.48760965298049735

Hyper-Parameter Tuning

TOP 16 Features Selected			
Random Forest (Estimator)		***Bagging Classifier***	
Hyper-Parameter	***Value***	***Hyper-Parameter***	***Value***
N Estimators	200	***N Estimators***	100
Max Depth	None	***N Jobs***	-1
Random State	42	***Random State***	42
Min Samples Split	2	***Bootstrap Features***	False
		Max Samples	1.0
		Max Features	1.0
		Bootstrap	True

Notes & insights

- When Applying Normal Random Forest lonely, got slightly inaccurate results than running with AdaBoost Algorithm. But when improve with Bagging rather that Boosting, it leads to better results like the table above.

Classification Model #4 (Draft): KNN Classification

Classification Results

MEASURE	***RESULT***

Accuracy	0.7067545304777595
Precision	0.5117004435667153
Recall	0.4745329108074206
F1-Score	0.47587125047902273

Hyper-Parameter Tuning

TOP 14 Features Selected	
KNN	
Hyper-Parameter	***Value***
N Neighbors	6
Gamma	1
Weights	'uniform'

Notes & insights

- Tried to apply **adaboost classifier**, but we found out after a lot of trials that **KNN** doesn't work with boosting algorithm due to it does not have the same weight penalty algorithm that the adaboost apply because its a lazy learner.

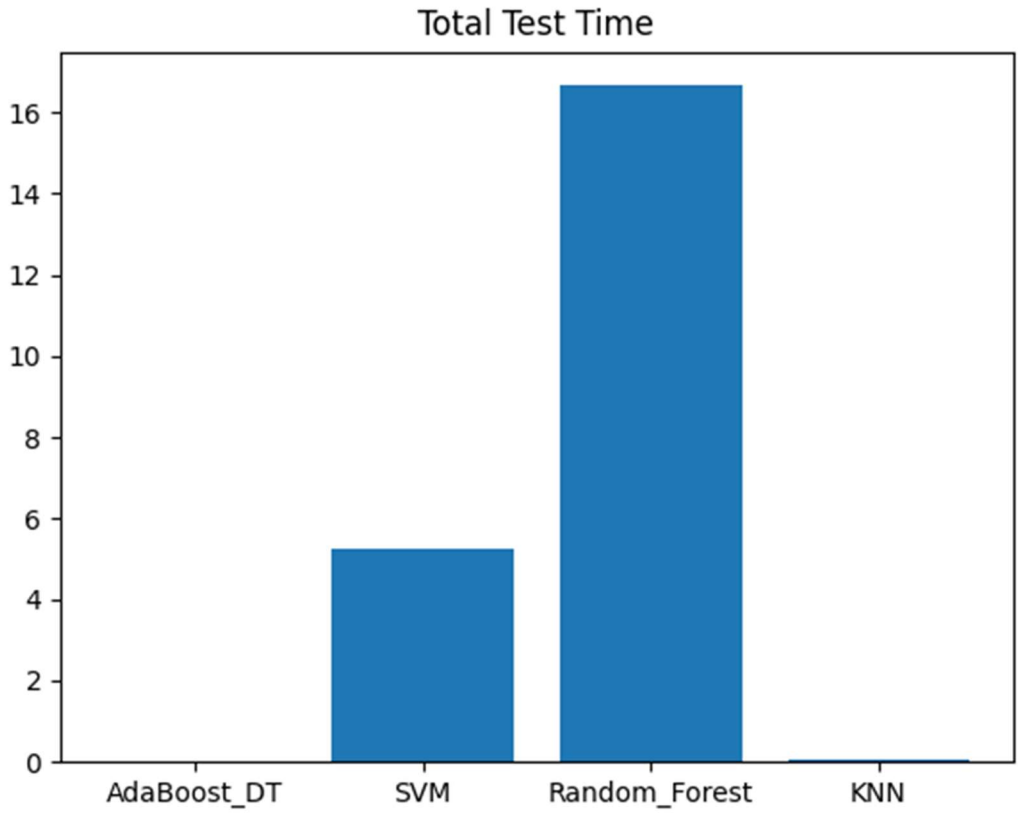
- we also tried to apply **BaggingClassifier**, but it didn't get us better results because the nature of the **KNN** algorithm doesn't benefit of the **BaggingClassifier** algorithm.

Estimated Time

Total Training Time

- **Decision Tree Classification:** 7.181988000869751 secs
- **SVM Classification:** 44.6864492893219 secs
- **Random Forest Classification:** 44.379764795303345 secs
- **KNN Classification:** 0.00797891616821289 secs

Total Testing Time



- **Decision Tree Classification:** 0.020943880081176758 secs
- **SVM Classification:** 5.274652719497681 secs
- **Random Forest Classification:** 16.665528059005737 secs
- **KNN Classification:** 0.041399478912353516 secs

Conclusion

We found out that pre-processing methods has paid off and cleaned the data properly.

The ANOVA feature selection method was the best method in our case as it said in lab 6, so we used it, and we used different number of features depending on the classification model used.

Picking classification method, we found out that SVM (1vs1) was the best model over the other 3 models, after applying bagging on it, it was even better and got the best results over all in the project.

In the end we Saved the models using pickle library, calculate estimated time, and prepared the project for taking new test sample.