Assignment 1 Report

ADARSHA MONDAL (MDS202205) | SAYANTIKA SENGUPTA (MDS202232)

TASK 1 (UCI Bank Marketing Dataset)

Dataset Description

The **Bank Marketing Dataset** from the UCI Machine Learning Repository is related with direct marketing campaigns (phone calls) of a Portuguese banking institution.

The classification goal is to predict if the client will subscribe a term deposit (variable `y`)

Explanatory variables (`x`):

- Bank client data:
  1. `age` (numeric)
  2. `job` : type of job (categorical: "admin.", "blue-collar", "entrepreneur", "housemaid", "management", "retired", "self-employed", "services", "student", "technician", "unemployed", "unknown"
  3. `marital` : marital status (categorical: "divorced", "married", "single", "unknown"; note: "divorced" means divorced or widowed)
  4. `education` (categorical: "basic.4y","basic.6y","basic.9y","high.school","illiterate","professional.course","university.degree","unknown")
  5. `default` : has credit in default? (categorical: "no", "yes", "unknown")
  6. `housing` : has housing loan? (categorical: "no", "yes", "unknown")
  7. `loan` : has personal loan? (categorical: "no", "yes", "unknown")
- Related with the last contact of the current campaign:
  8. `contact` : contact communication type (categorical: "cellular", "telephone")
  9. `month` : last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
  10. `day_of_week` : last contact day of the week (categorical: "mon","tue","wed","thu","fri")
  11. `duration` : last contact duration, in seconds (numeric). Important note: this attribute highly
- Other attributes:
  12. `campaign` : number of contacts performed during this campaign and for this client (numeric, includes last contact)
  13. `pdays` : number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
  14. `previous` : number of contacts performed before this campaign and for this client (numeric)
  15. `poutcome` : outcome of the previous marketing campaign (categorical: "failure", "nonexistent", "success")
- Social and economic context attributes:
  16. `emp.var.rate` : employment variation rate - quarterly indicator (numeric)
  17. `cons.price.idx` : consumer price index - monthly indicator (numeric)
  18. `cons.conf.idx` : consumer confidence index - monthly indicator (numeric)
  19. `euribor3m` : euribor-3 month rate - daily indicator (numeric)
  20. `nr.employed` : number of employees - quarterly indicator (numeric)

Output variable (desired target `y`):

21. `y` - has the client subscribed a term deposit? (binary: *yes*, *no*)

> **In conclusion:** Dataset has 11 Categorical, 5 Discrete Numerical and 5 Continuous Numerical features

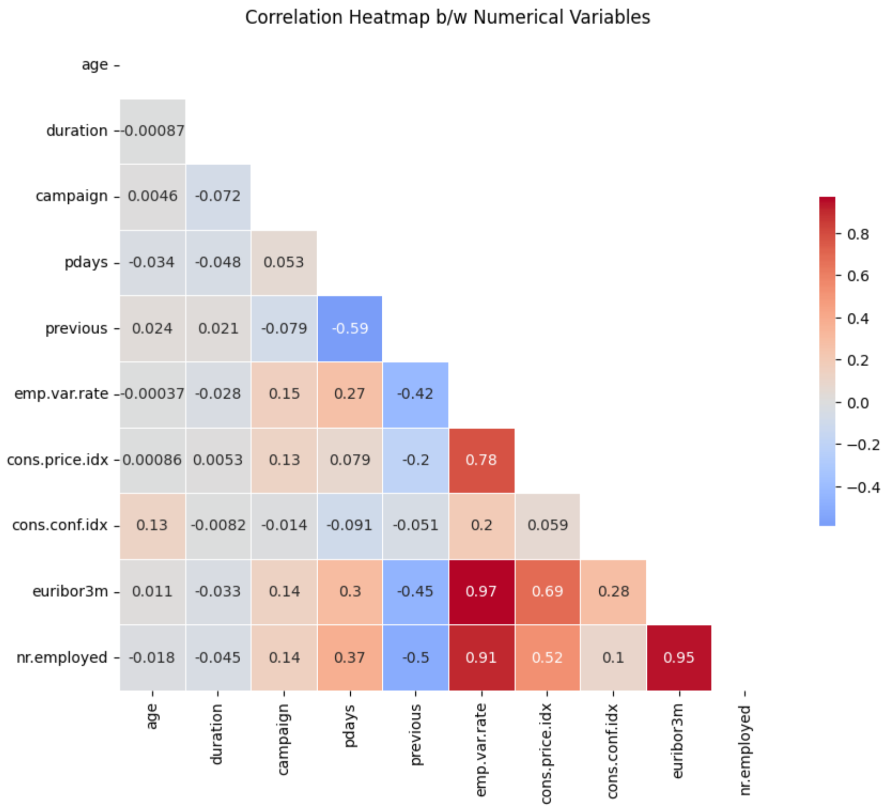
> **Missing Attribute Values**: There are several missing values in some categorical attributes, all coded with the *unknown* label. These missing values are treated using deletion or imputation techniques.

Description of numerical features

age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600

age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000

Collinearity among the Numerical variables



> **Insight**
> Variables like `euribor3m`, `nr.employed` and `emp.var.rate` have a correlation factor **>0.9** with each other. Which suggest that, these variables are very strongly correlated. Therefore keeping all of them in the dataset increases collinearity among explanatory variables which increases variance and thus leads to **overfitted** model.
>
> As we know, Decision Tree models are highly prone to overfitting. Thus decreasing collinearity happens to be one of the most significant preprocessing for optimal Decision Tree model building.

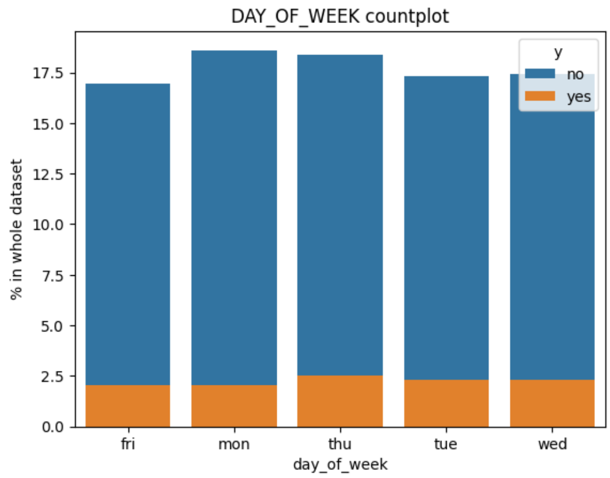
▼ Cleaning Data

▼ FEATURES

- `duration`

> This attribute highly affects the output target (e.g., if `duration`=*0* then `y`=*no*). Yet, the `duration` is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

- `day_of_week`



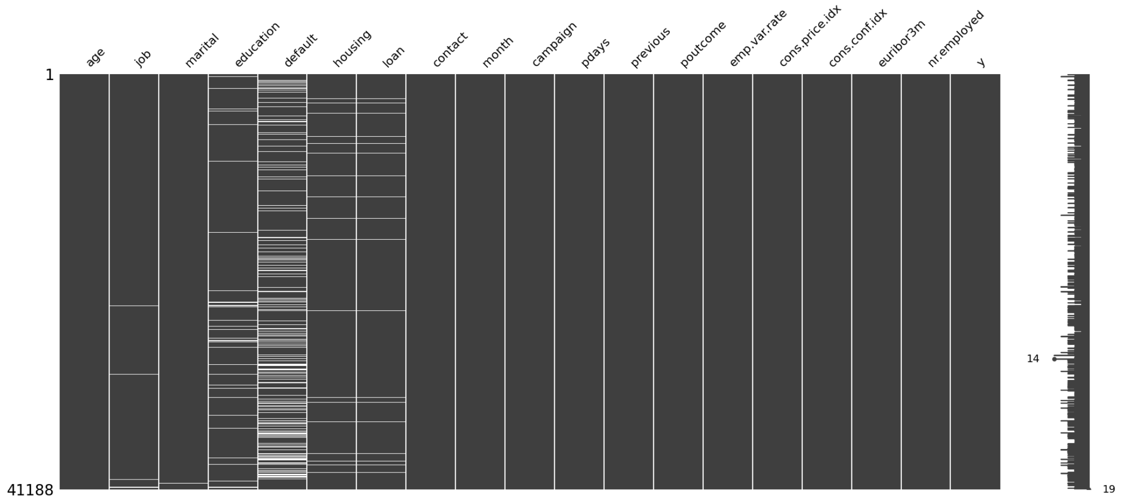
> Labels are uniformly distributed throughout the categories of `day_of_week`. So dropping this column would not result significant, if any information loss.

▼ **RECORDS**

> Some of the categorical features consists of the value *unknown*. Unless we claim it to be a separate category, it can be seen as missing values and we could find, it is distributed through multiple attributes in the following magnitude.

▼ **Missing Value Percentage before cleaning**

feature	unique_value_count	percent_missing (%)
`age`	78	0
`job`	12	0.8
`marital`	4	0.2
`education`	8	4.2
`default`	3	20.9
`housing`	3	2.4
`loan`	3	2.4
`contact`	2	0
`month`	10	0
`day_of_week`	5	0
`duration`	1544	0
`campaign`	42	0
`pdays`	27	0
`previous`	8	0
`poutcome`	3	0
`emp.var.rate`	10	0
`cons.price.idx`	26	0
`cons.conf.idx`	26	0
`euribor3m`	316	0
`nr.employed`	11	0
`y`	2	0



▼ **Missing Value Percentage after cleaning**

> After dropping those records as stated above, the only feature having missing values is `default` and the proportion of that is **20.3%.** Which we can impute later.

feature	percent_missing (%)
`age`	0
`job`	0
`marital`	0
`education`	0
`default`	20.3
`housing`	0
`loan`	0
`contact`	0
`month`	0
`day_of_week`	0
`duration`	0
`campaign`	0
`pdays`	0
`previous`	0
`poutcome`	0
`emp.var.rate`	0
`cons.price.idx`	0
`cons.conf.idx`	0
`euribor3m`	0
`nr.employed`	0
`y`	0

▼ Data Transformation

▼ Imputation

Proportion of class labels ("yes" and "no") within overall dataset and missing value records of `default` attriibute.

	no	yes
overall	0.888665	0.111335
`default`	0.948563	0.0514374

> The class imbalance is significantly more in the missing value records of `default` attribute, compared to overall dataset. We can leverage this fact and impute the missing values by applying **mode** strategy.

▼ Encoding

▼ **Ordinal Encoding**

`education`

Ordinally encoded `education` attribute by:

category	encode_to
`professional.course`	4
`university.degree`	3
`high.school`	2
`basic.6y`	1
`basic.9y`	1
`basic.4y`	1
`illiterate`	0

> **Notice:** On contrary to the original dataset, all the basic categories are but into one categories. And *university.degree (3)* has a lower encoding than *professional.course (4).*

Dummies Encoding

Remaining categorical features are Dummies encoding technique. And after encoding the features of the dataset will be the following:

`age`, `education`, `campaign`, `pdays`, `previous`, `emp.var.rate`, `cons.price.idx`, `cons.conf.idx`, `euribor3m`, `nr.employed`, `job_blue-collar`, `job_entrepreneur`, `job_housemaid`, `job_management`, `job_retired`, `job_self-employed`, `job_services`, `job_student`, `job_technician`, `job_unemployed`, `marital_married`, `marital_single`, `default_yes`, `housing_yes`, `loan_yes`, `contact_telephone`, `month_aug`, `month_dec`, `month_jul`, `month_jun`, `month_mar`, `month_may`, `month_nov`, `month_oct`, `month_sep`, `poutcome_nonexistent`, `poutcome_success`

Binning

`age`

binning_condition	bin_category
$x < 21$	0
$21 \leq x \leq 30$	1
$30 < x \leq 40$	2
$40 < x \leq 50$	3
$50 < x \leq 60$	4
$x > 60$	5

Normalization

From the data description of numerical features we can observe that some of them have negative values in them. This would cause error during **Naive Bayes classification**. Therefore we normalize all the continuous numerical features.

Fitting Model

Why Weighted Recall as evaluation metric ?

Our target variable (`y`) signify that, after reaching out to bank customer with the term deposit product is the customer subscribing to it or not.

Predicted/ Actual	*yes (True)*	*no (False)*
yes (True)	Predicted would subscribe and actually subscribes	Predicted won't subscribe but subscribes
no (False)	Predicted would subscribe but does not subscribe	Predicted won't subscribe and actually does not subscribe

> Now by logic, as bank is a profit making firm, and profit is correlated with revenue, therefore any kind of prospective loss of revenue (i.e. subscription to term deposit by a customer) is a business loss and thus detrimental to bank balance sheet.

> Therefore instead of **accuracy** we choose **recall,** more precisely **weighted recall** (as the the dataset is highly imbalanced) as the performance evaluation metric of the model.

Train-Test Split

The data is highly imbalanced i.e. ***no*** and ***yes*** labels have the proportion of **~89%** and **~11%,** respectively.

Therefore, we have to make a **stratified** split of the data into test and train datasets.

▼ MODEL 1 ~ Decision Tree Classifier

▼ Hyperparameter Tuning and Cross Validation

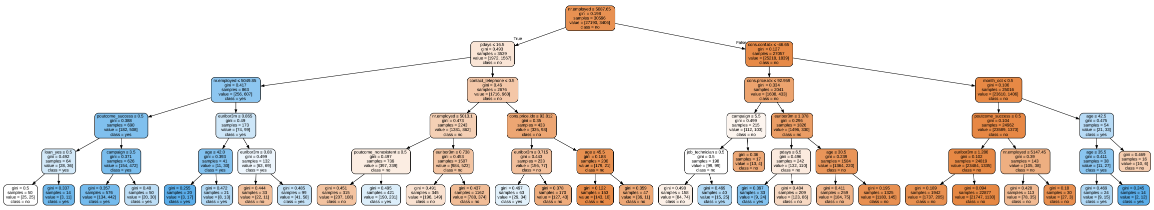
While fitting the training data we iterated through all of the combinations of hyperparameters as below. Also, do a **5 fold cross validation** for each combination of parameters:

hyperparameter	values
criterion	'gini','entropy'
max_depth	2, 3, 4, 5
min_samples_split	2, 3, 4, 5, 6, 8
min_samples_leaf	5, 10, 12, 15, 20, 30

After running the **GridSearchCV** on the train dataset we tune on the best parameters as:

parameter	best_parametric_value
criterion	***'gini'***
max_depth	***5***
min_samples_split	***2***
min_samples_leaf	***20***

▼ Graphics of the Decision Tree



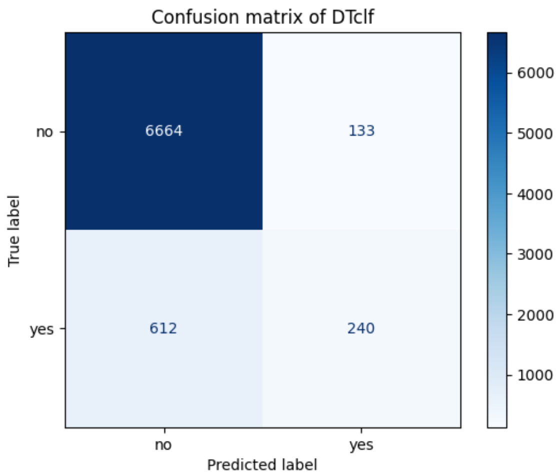
Decision Tree of Task 1

Best Train Score (Weighted Recall)

0.901719

Best Test Score (Weighted Recall)

0.902602



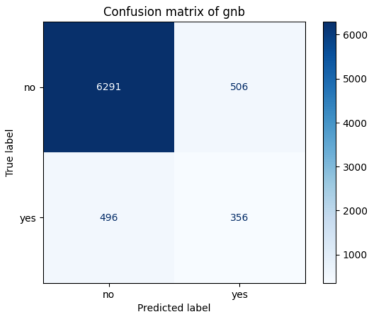
▼ MODEL 2 ~ Naive Bayes Classifier

We fit our dataset in 3 common Naive Bayes classification models, i.e. **Gaussian NB, Multimodal NB, Bernoulli NB.**

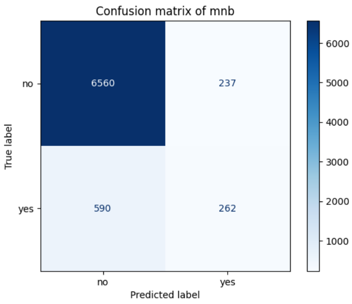
Cross Validation

For each NB model we do **5 fold cross validation** on the training dataset and calculate the **weighted recall** for each fold. The mean cross validation score will be taken as the test score of the model.

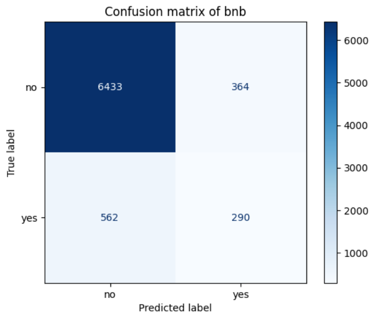
Performance Evaluation



Gaussian NB



Multinomial NB



Binomial NB

	TRAIN SCORE	TEST SCORE
Gaussian NB	0.870637	0.869002
Multimodal NB	0.889071	0.891881
Bernoulli NB	0.876651	0.878938

> Comparing every models of Naive Bayes **Multimodal NB** gives us the best result in terms of both Train and Test scores.

▼ Compare both models on other metrics

▼ COLUMNS

Represent different metrices to measure the models

- `train_whtd_rcl` **:** Training weighted recall
- `test_whtd_rcl` **:** Testing weighted recall
- `test_acc` **:** Test accuracy
- `test_prec` **:** Test precision
- `test_roc` **:** Test ROC AUC score
- `time_to_train(ns)` **:** Model training time in nanoseconds
- `model_size(KB)` **:** Model size on disk in KB

▼ INDECES

Represent different models

- `DTclf` **:** Decision Tree Classifier
- `gnb` **:** Gaussian NB
- `mnb` **:** Multinomial NB
- `bnb:` Binomial NB

	train_whtd_rcl	test_whtd_rcl	test_acc	test_prec	test_roc	time_to_train(ns)	model_size(KB)
DTclf	0.901719	0.902602	0.902602	0.643432	0.631061	6.80573e+07	5.63574
gnb	0.870637	0.869002	0.869002	0.412993	0.671698	3.07792e+07	2.31738
mnb	0.889071	0.891881	0.891881	0.52505	0.636322	1.21876e+07	2.33496
bnb	0.876651	0.878938	0.878938	0.443425	0.643411	2.26029e+07	2.35254

TASK 2 (Bollywood Movies Dataset)

▼ Dataset Description

The **Bollywood Movies Dataset** is a database of 1698 Hindi Movies from 2005-2017.

A movie is a *hit* if **revenue > budget**, and it is a *flop* otherwise. The goal is to predict whether a movie will be a *hit* or *flop*, given all the other attributes.

Once again, the task is to build two classifiers for this data set: a Decision Tree and a Naïve Bayes classifier.

▼ Explanatory variables (`x`):

1. `Movie Name` :(categorical: 'Golden Boys' 'Kaccha Limboo', 'Not A Love Story', 'Dunno Y Na Jaane Kyun', 'Taj Mahal - An Eternal Love Story'…)
2. `Release Period` : In which type of day it had been released (categorical: 'Normal', 'Holiday')

3. `Whether Remake`: Has it has a movie with the same name and synopsis(categorical: 'No', 'Yes')
4. `Whether Franchise`: Whether part of a production brand (categorical: 'No', 'Yes')
5. `Genre`: (categorical: 'suspense', 'drama', 'thriller', 'adult', 'comedy', 'action', 'love_story' , 'rom__com', 'horror', 'fantasy', 'masala', 'mythological', 'animation', 'documentary')
6. `New Actor`: Whether the lead actor's first movie (categorical: 'No', 'Yes')
7. `New Director`: Whether the Director's first movie (categorical: 'No', 'Yes')
8. `New Music Director`: Whether the Music Director's first movie (categorical: 'No', 'Yes')
9. `Lead Star`: Name of lead actor (categorical: 'Jeet Goswami', 'Karan Bhanushali', 'Mahie Gill', 'Aadar Jain', 'Aadil Khan'…)
10. `Director`: Name of the director (categorical: 'Ravi Varma', 'Sagar Ballary', 'Ram Gopal Verma' ... 'Vikram Chopra','Sanghamitra Chaudhuri', 'Akbar Khan')
11. `Music Director`: Name of the music director (categorical: 'Baba Jagirdar', 'Amardeep Nijjer', 'Sandeep Chowta', 'Amit Trivedi', 'Babloo Ustad',…)
12. `Number of Screens`: Count of screens the movie has been released on (discrete numerical)
13. `Revenue(INR)`: Total revenue of collection from all sources (discrete numerical)
14. `Budget(INR)`:(discrete numerical)

> By analysing real world box-office data we can corroborate that, the `Revenue(INR)` and `Budget(INR)` of the original dataframe need to be interchanged.

▼ Output variable (desired target `y`):

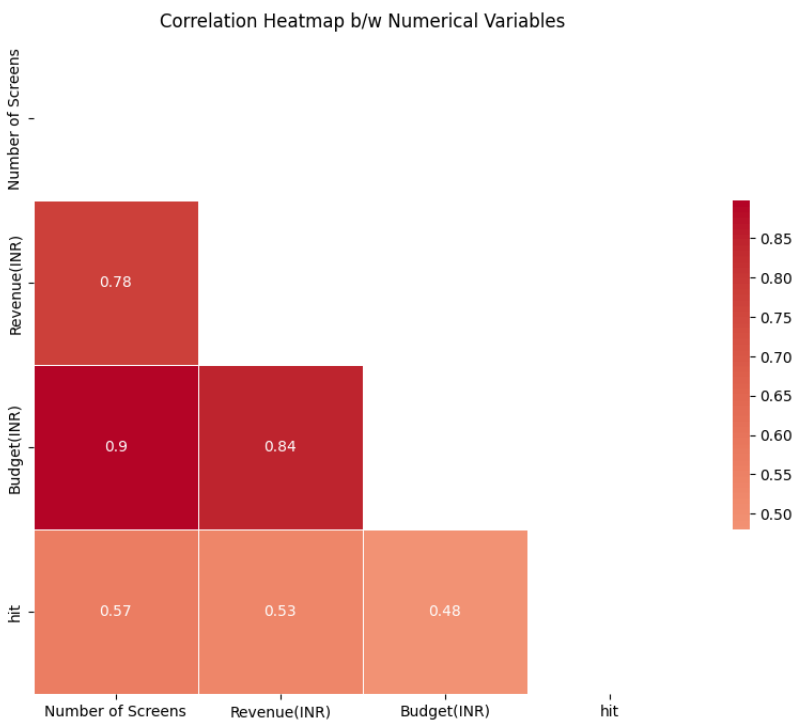
15. `hit`**:** Whether the movie hits (categorical: 'No', 'Yes')

> **In conclusion:** Dataset has 11 Categorical, 3 Discrete Numerical features.

▼ Description of numerical features

	Number of Screens	Revenue(INR)	Budget(INR)
count	1698	1698	1698
mean	553.832	2.37729e+08	1.50167e+08
std	782.952	6.1344e+08	2.43484e+08
min	1	7250	325000
25%	30	1.15e+06	1.5e+07
50%	200	1.24e+07	5.5e+07
75%	800	1.77832e+08	1.9e+08
max	4600	8.01612e+09	2.1e+09

Collinearity among the Numerical variables



> **Insight**
> Features like `Revenue(INR)` and `Budget(INR)` have a correlation factor **>0.8** with each other. Which suggest that, these variables are very strongly correlated. Therefore keeping all of them in the dataset increases collinearity among explanatory variables which increases variance and thus leads to **overfitted** model.
>
> As we know, Decision Tree models are highly prone to overfitting. Thus decreasing collinearity happens to be one of the most significant preprocessing for optimal Decision Tree model building.

▼ Cleaning Data

▼ FEATURES

- `Revenue(INR)`

  > This attribute highly affects the output target (e.g., if `Budget(INR)` < `Revenue(INR)` then `y` =*no*). Yet, the `Revenue(INR)` is not known before a movie's box office collection completes. Also, after including this `hit` is obviously known. Thus, this feature should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

- we can see `Movie Name` has nearly equal no. of categories as the no. of records. Therefore keeping this feature would lead to high variance.

  Similarly, other categorical features like `Lead Star`, `Director` & `Music Director` has unique categories **>33%** of total records count. As Decision Tree Classifier model is **prone to overfitting**, therefore dropping these features would reduce the complexity of the model and thus stop it from overfitting.

> After dropping some features, remaining features: `Release Period`, `Whether Remake`, `Whether Franchise`, `Genre`, `New Actor`, `New Director`, `New Music Director`, `Number of Screens`, `Budget(INR)`

▼ RECORDS

> There is no missing values, so no need to drop any record.

▼ Data Transformation

▼ Encoding

▼ Dummies Encoding

Remaining categorical features are transformed Dummies encoding technique. And after encoding the features of the dataset will be the following:

`Number of Screens` , `Budget(INR)` , `Release Period_Normal` , `Whether Remake_Yes` , `Whether Franchise_Yes` , `Genre_adult` , `Genre_animation` , `Genre_comedy` , `Genre_documentary` , `Genre_drama` , `Genre_fantasy` , `Genre_horror` , `Genre_love_story` , `Genre_masala` , `Genre_mythological` , `Genre_rom__com` , `Genre_suspense` , `Genre_thriller` , `New Actor_Yes` , `New Director_Yes` , `New Music Director_Yes`

▼ Fitting Model

▼ Why Weighted Recall as evaluation metric ?

Our target variable (`hit`) signify that giving these features and its values, whether it is going to hit or not

Predicted/ Actual	*yes (True)*	*no (False)*
yes (True)	Predicted would hit and actually hits	Predicted won't hit but hits
no (False)	Predicted would hit but does not hit	Predicted would not hit and actually does not hit

> Not buying a movie which my model predicted to be a flop but ended up being a once-in-a-decade hit movie could be more detrimental from a business POV.
>
> The reason is that business like the OTT platform has to pay a considerable amount of money to acquire the digital rights of a movie, and not buying a potentially successful movie may result in losing out on revenue and market share to competitors. In contrast, if the company buys a movie that ends up being a flop, the financial loss can be absorbed by the company as part of its risk management strategy.

> Therefore instead of **accuracy** we choose **recall,** more precisely **weighted recall** (as the the dataset is highly imbalanced) as the performance evaluation metric of the model.

▼ Train-Test Split

The data is highly imbalanced i.e. ***no*** and ***yes*** labels have the proportion of **~72%** and **~28%,** respectively.

Therefore, we have to make a **stratified** split of the data into test and train datasets.

▼ MODEL 1 ~ Decision Tree Classifier

▼ Hyperparameter Tuning and Cross Validation

For different cross validation folds (between **2 to 8**, both included) we fit the training data, by iterating through all of the combinations of hyperparameters as below. From this we select the best count of folds for which the model gives the maximum training score:

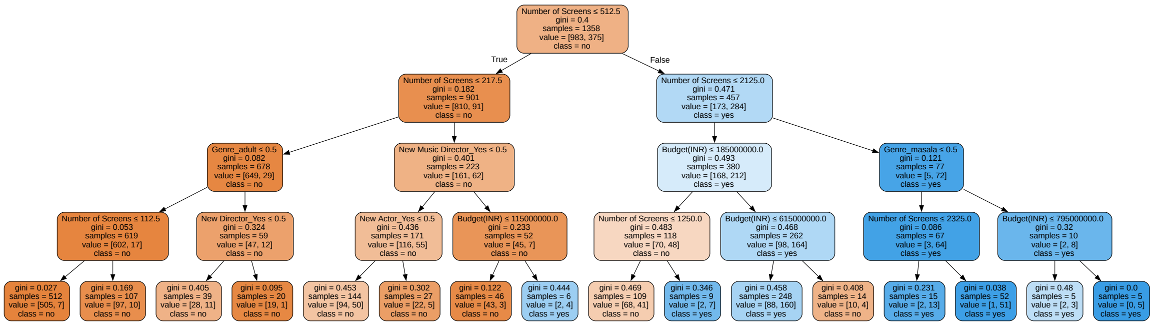
hyperparameter	values
criterion	'gini','entropy'
max_depth	2, 3, 4, 5
min_samples_split	1, 2, 3, 4, 5
min_samples_leaf	1, 2, 3, 4, 5, 10, 12

> The best count of fold comes out to be **5.**

After running the **GridSearchCV** on the train dataset with 5 fold CV and hyperparameter tuning, the best combination of hyperparameter comes out to be as below:

parameter	best_parametric_value
criterion	***'gini'***
max_depth	***4***
min_samples_split	***1***
min_samples_leaf	***5***

▼ Graphics of the Decision Tree



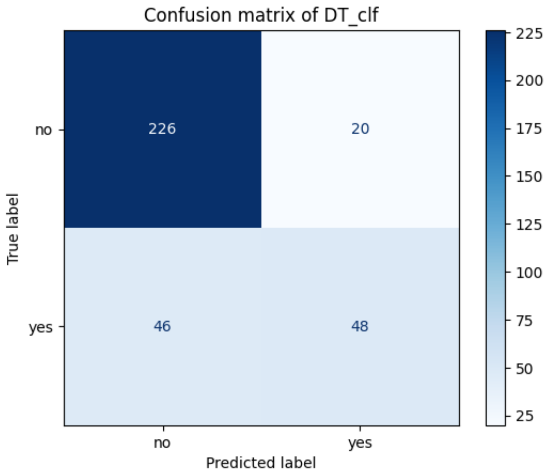
Decision Tree of Task 2

Best Train Score (Weighted Recall)

0.818125

Best Test Score (Weighted Recall)

0.805882



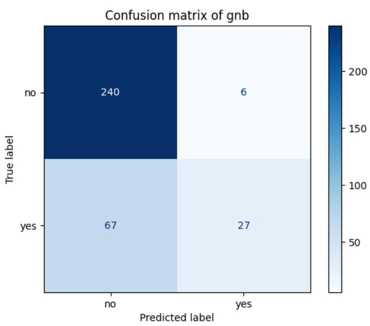
▼ MODEL 2 ~ Naive Bayes Classifier

We fit our dataset in 3 common Naive Bayes classification models, i.e. **Gaussian NB, Multimodal NB, Bernoulli NB.**

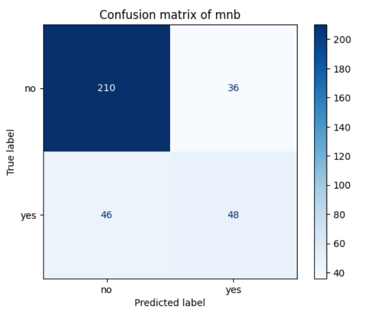
Cross Validation

For each NB model we do **5 fold cross validation** on the training dataset and calculate the **weighted recall** for each fold. The mean cross validation score will be taken as the test score of the model.

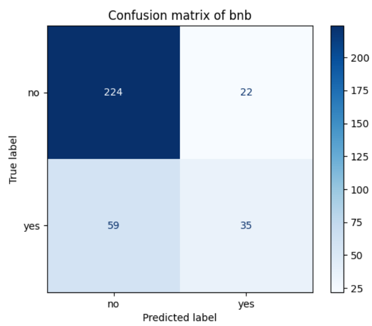
Performance Evaluation



Gaussian NB



Multinomial NB



Binomial NB

	TRAIN SCORE	TEST SCORE
Gaussian NB	0.784241	0.785294

	TRAIN SCORE	TEST SCORE
Multimodal NB	0.749677	0.758824
Bernoulli NB	0.722374	0.761765

> Comparing every models of Naive Bayes **Gaussian NB** gives us the best result in terms of both Train and Test scores.

▼ Compare both models on other metrics

▼ COLUMNS

Represent different metrices to measure the models

- `train_whtd_rcl` **:** Training weighted recall
- `test_whtd_rcl` **:** Testing weighted recall
- `test_acc` **:** Test accuracy
- `test_prec` **:** Test precision
- `test_roc` **:** Test ROC AUC score
- `time_to_train(ns)` **:** Model training time in nanoseconds
- `model_size(KB)` **:** Model size on disk in KB

▼ INDECES

Represent different models

- `DTclf` **:** Decision Tree Classifier
- `gnb` **:** Gaussian NB
- `mnb` **:** Multinomial NB
- `bnb:` Binomial NB

	train_whtd_rcl	test_whtd_rcl	test_acc	test_prec	test_roc	time_to_train(ns)	model_size(KB)
DTclf	0.818125	0.805882	0.805882	0.705882	0.714669	3.8615e+06	3.63965
gnb	0.784241	0.785294	0.785294	0.818182	0.631422	2.5081e+06	1.68262
mnb	0.749677	0.758824	0.758824	0.571429	0.682148	1.6352e+06	1.7002
bnb	0.722374	0.761765	0.761765	0.614035	0.641455	2.5861e+06	1.71777