Movie reviews Classification

Problem Statement - For Movie Audience it is hard to find which movie is good or bad without watching classification on basis of reviews will help decide audience whether they should watch movie or not

```
# Dataset link - https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews


# Libraries import

import numpy as np
import pandas as pd
import os


import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer


# downloading necessary dependencies

nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
# reading csv file

movie_reviews = pd.read_csv("IMDB Dataset.csv")


# checking data

movie_reviews
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
...	...	...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

50000 rows × 2 columns

```
# Checking null records
```

```
movie_reviews.isnull().sum()
```

```
review       0
sentiment    0
dtype: int64
```

```
# initializing stemming

ps= PorterStemmer()
```

```
# making corpus -removing secial characters, making lower case, removing stopwords and stemming

corpus = []
for i in range(0, len(movie_reviews)):
    review = re.sub('[^a-zA-Z0-9]', ' ', movie_reviews['review'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    print("no. of iterations :", i)
    review = ' '.join(review)
    corpus.append(review)
```

```
Streaming output truncated to the last 5000 lines.
no. of iterations : 45000
no. of iterations : 45001
no. of iterations : 45002
no. of iterations : 45003
no. of iterations : 45004
no. of iterations : 45005
no. of iterations : 45006
no. of iterations : 45007
no. of iterations : 45008
no. of iterations : 45009
no. of iterations : 45010
no. of iterations : 45011
no. of iterations : 45012
no. of iterations : 45013
no. of iterations : 45014
no. of iterations : 45015
no. of iterations : 45016
no. of iterations : 45017
no. of iterations : 45018
no. of iterations : 45019
no. of iterations : 45020
no. of iterations : 45021
no. of iterations : 45022
no. of iterations : 45023
no. of iterations : 45024
no. of iterations : 45025
no. of iterations : 45026
no. of iterations : 45027
no. of iterations : 45028
no. of iterations : 45029
no. of iterations : 45030
no. of iterations : 45031
no. of iterations : 45032
no. of iterations : 45033
no. of iterations : 45034
no. of iterations : 45035
no. of iterations : 45036
no. of iterations : 45037
no. of iterations : 45038
no. of iterations : 45039
no. of iterations : 45040
no. of iterations : 45041
no. of iterations : 45042
no. of iterations : 45043
no. of iterations : 45044
no. of iterations : 45045
no. of iterations : 45046
no. of iterations : 45047
no. of iterations : 45048
no. of iterations : 45049
no. of iterations : 45050
no. of iterations : 45051
no. of iterations : 45052
no. of iterations : 45053
no. of iterations : 45054
no. of iterations : 45055
```

```
no. of iterations : 45056
```

Challenges - As dataset was large - 50,000 sentences- it took more than 1.5 hour to stem all words, Used google colab - did in 10 minutes

```
# showing corpus

corpus
```

```
['one review mention watch 1 oz episod hook right exactli happen br br first thing struck oz brutal unflinch scene violenc set right word go trust show faint heart timid show pull punch regard drug sex violenc hardcor classic use word br br call oz nicknam given oswald maximum secur state penitentari focus mainli emerald citi experiment section prison cell glass front face inward privaci high agenda em citi home mani aryan muslim gangsta latino christian italian irish scuffl death stare dodgi deal shadi agreement never far away br br would say main appeal show due fact goe show dare forget pretti pictur paint mainstream audienc forget charm forget romanc oz mess around first episod ever saw struck nasti surreal say readi watch develop tast oz got accustom high level graphic violenc violenc injustic crook guard sold nickel inmat kill order get away well manner middl class inmat turn prison bitch due lack street skill prison experi watch oz may becom comfort uncomfort view that get touch darker side',
 'wonder littl product br br film techniqu unassum old time bbc fashion give comfort sometim discomfort sens realism entir piec br br actor extrem well chosen michael sheen got polari voic pat truli see seamless edit guid refer william diari entri well worth watch terrificli written perform piec master product one great master comedi life br br realism realli come home littl thing fantasi guard rather use tradit dream techniqu remain solid disappear play knowledg sens particularli scene concern orton halliwel set particularli flat halliwel mural decor everi surfac terribl well done',
 'thought wonder way spend time hot summer weekend sit air condit theater watch light heart comedi plot simplist dialogu witti charact likabl even well bread suspect serial killer may disappoint realiz match point 2 risk addict thought proof woodi allen still fulli control style mani us grown love br br laugh one woodi comedi year dare say decad never impress scarlet johanson manag tone sexi imag jump right averag spirit young woman br br may crown jewel career wittier devil wear prada interest superman great comedi go see friend',
 'basic famili littl boy jake think zombi closet parent fight time br br movi slower soap opera suddenli jake decid becom rambo kill zombi br br ok first go make film must decid thriller drama drama movi watchabl parent divorc argu like real life jake closet total ruin film expect see boogeyman similar movi instead watch drama meaningless thriller spot br br 3 10 well play parent descent dialog shot jake ignor',
 'petter mattei love time money visual stun film watch mr mattei offer us vivid portrait human relat movi seem tell us money power success peopl differ situat encount br br variat arthur schnitzler play theme director transfer action present time new york differ charact meet connect one connect one way anoth next person one seem know previou point contact stylishli film sophist luxuri look taken see peopl live world live habitat br br thing one get soul pictur differ stage loneli one inhabit big citi exactli best place human relat find sincer fulfil one discern case peopl encount br br act good mr mattei direct steve buscemi rosario dawson carol kane michael imperioli adrian grenier rest talent cast make charact come aliv br br wish mr mattei good luck await anxious next work',
 'probabl time favorit movi stori selfless sacrific dedic nobl caus preachi bore never get old despit seen 15 time last 25 year paul luka perform bring tear eye bett davi one truli sympathet role delight kid grandma say like dress midget children make fun watch mother slow awaken happen world roof believ startl dozen thumb movi',
 'sure would like see resurrect date seahunt seri tech today would bring back kid excit grew black white tv seahunt gunsmok hero everi week vote comeback new sea hunt need chang pace tv would work world water adventur oh way thank outlet like view mani viewpoint tv mani movi ole way believ got wanna say would nice read plu point sea hunt rhyme would 10 line would let submit leav doubt quit must go let',
 'show amaz fresh innov idea 70 first air first 7 8 year brilliant thing drop 1990 show realli funni anymor continu declin complet wast time today br br truli disgrac far show fallen write pain bad perform almost bad mildli entertain respit guest host show probabl still air find hard believ creator hand select origin cast also chose band hack follow one recogn brillianc see fit replac mediocr felt must give 2 star respect origin cast made show huge success show aw believ still air',
 'encourag posit comment film look forward watch film bad mistak seen 950 film truli one worst aw almost everi way edit pace storylin act soundtrack film song lame countri tune play less four time film look cheap nasti bore extrem rare happi see end credit film br br thing prevent give 1 score harvey keitel far best perform least seem make bit effort one keitel obsess',
 'like origin gut wrench laughter like movi young old love movi hell even mom like br br great camp',
 'phil alien one quirki film humour base around odd everyth rather actual punchlin br br first odd pretti funni movi progress find joke odd funni anymor br br low budget film that never problem pretti interest charact eventu lost interest br br imagin film would appeal stoner current partak br br someth similar better tri brother anoth planet',
 'saw movi 12 came recal scariest scene big bird eat men dangl helplessli parachut right air horror horror br br young kid go cheesi b film saturday afternoon still tire formula monster type movi usual includ hero beauti woman might daughter professor happi resolut monster die end care much romant angl 12 year old predict plot love unintent humor br br year later saw psycho came love star janet leigh bump earli film sat took notic point sinc screenwrit make stori make scari possibl well worn formula rule',
 'im big fan boll work mani enjoy movi postal mayb im one boll appar bought right use far cri long ago even game even finsish br br peopl enjoy kill merc infiltr secret research lab locat tropic island warn far cri someth mr boll scheme togeth along legion schmuck feel loneley set mr boll invit three countrymen play player go name til schweiger udo kier ralf moeller br br three name actual made self pretti big movi biz tale goe like jack carver play til schweiger ye carver german hail bratwurst eat dude howev find til act movi pretti badass peopl complain realli stay true whole carver agenda saw carver first person perspect realli know look like kick br br howev storylin film beyond dement see evil mad scientist dr krieger play udo kier make genet mutat soldier gm call perform top secret research island remind spoiler vancouv reason that right palm tree instead got nice rich lumberjack wood even gone far start cri meheh cannot go wanna stay true boll shenanigan go see movi disappoint deliv true boll experi mean suck br br thing worth mention
```

```
# saving corpus in to dataframe

corpus_data= pd.DataFrame(corpus,columns=['review'])


# adding y column in to corpus_data dataframe

corpus_data['sentiment']=movie_reviews['sentiment']


# checking corpus_dat
```

```
corpus_data
```

	review	sentiment
0	one review mention watch 1 oz episod hook righ...	positive
1	wonder littl product br br film techniqu unass...	positive
2	thought wonder way spend time hot summer weeke...	positive
3	basic famili littl boy jake think zombi closet...	negative
4	petter mattei love time money visual stun film...	positive
...	...	...
49995	thought movi right good job creativ origin fir...	positive
49996	bad plot bad dialogu bad act idiot direct anno...	negative
49997	cathol taught parochi elementari school nun ta...	negative
49998	go disagre previou comment side maltin one sec...	negative
49999	one expect star trek movi high art fan expect ...	negative

50000 rows × 2 columns

```
# saving data in to csv file

# corpus_data.to_csv('corpus.txt')


# defining 'X' independent feature

X= corpus_data['review']
X
```

```
0        one review mention watch 1 oz episod hook righ...
1        wonder littl product br br film techniqu unass...
2        thought wonder way spend time hot summer weeke...
3        basic famili littl boy jake think zombi closet...
4        petter mattei love time money visual stun film...
                              ...
49995    thought movi right good job creativ origin fir...
49996    bad plot bad dialogu bad act idiot direct anno...
49997    cathol taught parochi elementari school nun ta...
49998    go disagre previou comment side maltin one sec...
49999    one expect star trek movi high art fan expect ...
Name: review, Length: 50000, dtype: object
```

```
#defining label feature

y = corpus_data['sentiment']
y
```

```
0        positive
1        positive
2        positive
3        negative
4        positive
           ...
49995    positive
49996    negative
49997    negative
49998    negative
49999    negative
Name: sentiment, Length: 50000, dtype: object
```

```
# changing y column values in to numerical values

y=pd.get_dummies(corpus_data['sentiment'],drop_first=True,dtype = int)
y
```

```
        positive
0           1
1           1
2           1
3           0
4           1
...       ...
49995       1
49996       0
49999       0
```

```
# train test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

```
X_train,y_train
```

```
(23990    randolph scott head albuquerqu take job uncl h...
 8729     like movi caus good approach buddhism exampl w...
 3451     well expect anyth deep meaning fight scene pre...
 2628     realli deserv rate even neg ten watch show age...
 38352    dwight frye steal show one foolish young man s...
                               ...
 11284    shadow magic recaptur joy amaz first movi audi...
 44732    found movi quit enjoy fairli entertain good ch...
 38158    avoid one terribl movi excit pointless murder ...
 860      product quit surpris absolut love obscur earli...
 15795    decent movi although littl bit short time pack...
 Name: review, Length: 33500, dtype: object,
        positive
 23990         1
 8729          1
 3451          1
 2628          0
 38352         1
 ...         ...
 11284         1
 44732         1
 38158         0
 860           1
 15795         1

 [33500 rows x 1 columns])
```

```
X_test,y_test
```

```
(33553    realli like summerslam due look arena curtain ...
 9427     mani televis show appeal quit mani differ kind...
 199      film quickli get major chase scene ever increa...
 12447    jane austen would definit approv one br br gwy...
 39489    expect somewhat high went see movi thought ste...
                               ...
 27615    id like say film good touch film explain real ...
 21964    fulci one time favorit italian splatter direct...
 33321    case like movi saw found expect mean one anim ...
 40225    realli like movi thought entertain well dramat...
 28203    saw movi day ago hell br br like movi brian ha...
 Name: review, Length: 16500, dtype: object,
        positive
 33553         1
 9427          1
 199           0
 12447         1
 39489         0
 ...         ...
 27615         1
 21964         1
 33321         1
 40225         1
 28203         0

 [16500 rows x 1 columns])
```

Bag of words Technique

```
# Creating the Bag of Words model

cv = CountVectorizer(max_features=2500,binary = True,ngram_range=(1,2))
X_train_vec = cv.fit_transform(X_train).toarray()
X_test_vec =  cv.transform(X_test).toarray()


# checking X_train_vec

X_train_vec.shape
```

```
(33500, 2500)
```

```
# checking X_test_vec

X_test_vec.shape
```

```
(16500, 2500)
```

```
y_train,y_test
```

```
(       positive
 23990         1
 8729          1
 3451          1
 2628          0
 38352         1
 ...         ...
 11284         1
 44732         1
 38158         0
 860           1
 15795         1

 [33500 rows x 1 columns],
        positive
 33553         1
 9427          1
 199           0
 12447         1
 39489         0
 ...         ...
 27615         1
 21964         1
 33321         1
 40225         1
 28203         0

 [16500 rows x 1 columns])
```

Model Defining and training

```
#pip install catboost


# importing ML Libraries
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
# from catboost import CatBoostClassifier
import xgboost as xgb
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB

#importing metrics score libraries
from sklearn.metrics import (accuracy_score,
                             confusion_matrix,
                             classification_report,roc_auc_score)
from sklearn.metrics import precision_score,recall_score
```

```
# Logistic Regression classsification

LR = LogisticRegression(penalty='l2',C=0.001,max_iter=1000)


# Classification model using decision tree

DT= DecisionTreeClassifier()


# Classification model using Random Forest classifier

RF =RandomForestClassifier()


# Classification model using Ada boost classifier

AB=AdaBoostClassifier()


# XGBoost Model

XG = xgb.XGBClassifier(n_estimators=20,max_depth=10)


# # CatBoost Model

# CB=CatBoostClassifier()


# Gradient-Boost Model

GB=GradientBoostingClassifier()


# Support vector classification Model

SV = SVC(probability=True)


# K-Nearest Neighbour Model

KNN= KNeighborsClassifier()


Naive_bayes = MultinomialNB()


# model accuracy, predcision score and classification report in Bag of words

Model_list ={"Logistic Regression":LR,"Decision Tree": DT,
             "Random Forest":RF,"K Nearest Neigbour": KNN,"Naive bayes":Naive_bayes}

auc_report = {}
precision_report ={}

for model_name,model in Model_list.items():
    # pipe=make_pipeline(ct,ct2,model)    # making pipeline of tranformers

    # gs= GridSearchCV(pipe,cv=10,param_grid=param,n_jobs=-1,scoring='accuracy')
    # gs.fit(X_train_os,y_train_os)
    # pipe.set_params(**gs.best_params_)
    model.fit(X_train_vec, y_train)                              # ftting pipeline to train data


    y_pred_on_train=model.predict(X_train_vec)                    # predicting y_pred from training data
    y_pred_on_test=model.predict(X_test_vec)                        # predicting y_pred from test data
    y_pred_prob_test =model.predict_proba(X_test_vec)[:,1]            # predicting y_pred probability from test data
    # n_errors_train =  (y_pred_on_train!=y_train).sum()        # no. of errors in training data
    # n_errors_test =  (y_pred_on_test!=y_test).sum()              # no. of errors in  testing data
    training_report = classification_report(y_train, y_pred_on_train,output_dict=True)  # classification report from training data
    df_classification_report_train = pd.DataFrame(training_report).transpose()             # converting report to dataframe
    testing_report = classification_report(y_test, y_pred_on_test,output_dict=True)        # clasification report from testing data
    df_classification_report_test = pd.DataFrame(testing_report).transpose()               # converting report to dataframe
    auc_score_test=roc_auc_score(y_test, y_pred_prob_test)                                 # auc score on testing data
    auc_report[model_name]= auc_score_test
    precision = precision_score(y_test, y_pred_on_test)
    precision_report[model_name] =precision
```

```
    print(f"-----Metrics score of {model_name}----- \n")        #printing model name
    # print(f"best params K :{gs.best_params_}")
    # print(f"errors on training data {n_errors_train} : \n")     # printing errors on training data
    print(f"Accuracy Score on training data : {accuracy_score(y_train, y_pred_on_train)}") # printing accuracy score on train data
    print(f"Confusion Matrix on training data : \n {confusion_matrix(y_train, y_pred_on_train)}") #training data confusion matrix
    print(f"Classification Report on training data :\n {df_classification_report_train}\n")         # classification report on train data


    print("\n")     # next line
    # print(f"errors on testing data {n_errors_test} : \n")   # printing errors on testing data
    print(f"Accuracy Score on test data : {accuracy_score(y_test, y_pred_on_test)}\n")  # printing accuracy score on testing data
    print(f"Confusion Matrix on testing data : \n {confusion_matrix(y_test, y_pred_on_test)}") #training data confusion matrix
    print(f"Classification Report on testing data : \n {df_classification_report_test}\n")   # classification report printing
    print(f"Auc_score on testing data : \n {auc_score_test}")  #auc score printing
    print(f"Precision Score on test data : {precision_score(y_test, y_pred_on_test)}\n") # precision score on test data



best_model_auc = max(auc_report.values())
best_model = [i for i,j in auc_report.items() if j==best_model_auc]
print(f"best model is :  {best_model} with auc score {best_model_auc}")

best_model_precision = max(precision_report.values())
best_model_prec_name = [i for i,j in precision_report.items() if j==best_model_precision]
print(f"best model_precision is :  {best_model_prec_name} with precision score {best_model_precision}")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1
  y = column_or_1d(y, warn=True)
-----Metrics score of Logistic Regression-----

Accuracy Score on training data : 0.8622089552238806
Confusion Matrix on training data :
 [[14149  2643]
 [ 1973 14735]]
Classification Report on training data :
               precision    recall  f1-score       support
0              0.877621  0.842604  0.859756  16792.000000
1              0.847911  0.881913  0.864578  16708.000000
accuracy       0.862209  0.862209  0.862209      0.862209
macro avg      0.862766  0.862258  0.862167  33500.000000
weighted avg   0.862803  0.862209  0.862161  33500.000000



Accuracy Score on test data : 0.8541212121212122

Confusion Matrix on testing data :
 [[6827 1381]
 [1026 7266]]
Classification Report on testing data :
               precision    recall  f1-score       support
0              0.869349  0.831750  0.850134   8208.000000
1              0.840291  0.876266  0.857902   8292.000000
accuracy       0.854121  0.854121  0.854121      0.854121
macro avg      0.854820  0.854008  0.854018  16500.000000
weighted avg   0.854746  0.854121  0.854038  16500.000000

Auc_score on testing data :
 0.9315515336184432
Precision Score on test data : 0.8402914305539494

-----Metrics score of Decision Tree-----

Accuracy Score on training data : 1.0
Confusion Matrix on training data :
 [[16792     0]
 [    0 16708]]
Classification Report on training data :
               precision  recall  f1-score  support
0                    1.0     1.0       1.0  16792.0
1                    1.0     1.0       1.0  16708.0
accuracy             1.0     1.0       1.0      1.0
macro avg            1.0     1.0       1.0  33500.0
weighted avg         1.0     1.0       1.0  33500.0
```

```
Accuracy Score on test data : 0.7123030303030303

Confusion Matrix on testing data :
 [[5873 2335]
 [2412 5880]]
Classification Report on testing data :
```

TFIDF Approach

```
# Creating the TFIDF model
tv = TfidfVectorizer(max_features = 5000, ngram_range=(1,2))
X_train_vec = tv.fit_transform(X_train).toarray()
X_test_vec =  tv.transform(X_test).toarray()
```

```
X_train_vec.shape
```

```
(33500, 5000)
```

```
X_test_vec.shape
```

```
(16500, 5000)
```

```
# model accuracy, precision score and classification report in TFIDF

Model_list ={"Logistic Regression":LR,"Decision Tree": DT,
             "Random Forest":RF,"K Nearest Neigbour": KNN,"Naive bayes":Naive_bayes}

auc_report = {}
precision_report ={}

for model_name,model in Model_list.items():
    # pipe=make_pipeline(ct,ct2,model)    # making pipeline of tranformers

    # gs= GridSearchCV(pipe,cv=10,param_grid=param,n_jobs=-1,scoring='accuracy')
    # gs.fit(X_train_os,y_train_os)
    # pipe.set_params(**gs.best_params_)
    model.fit(X_train_vec, y_train)                              # ftting pipeline to train data

    y_pred_on_train=model.predict(X_train_vec)                     # predicting y_pred from training data
    y_pred_on_test=model.predict(X_test_vec)                         # predicting y_pred from test data
    y_pred_prob_test =model.predict_proba(X_test_vec)[:,1]           # predicting y_pred probability from test data
    # n_errors_train =  (y_pred_on_train!=y_train).sum()        # no. of errors in training data
    # n_errors_test =  (y_pred_on_test!=y_test).sum()              # no. of errors in  testing data
    training_report = classification_report(y_train, y_pred_on_train,output_dict=True)  # classification report from training data
    df_classification_report_train = pd.DataFrame(training_report).transpose()             # converting report to dataframe
    testing_report = classification_report(y_test, y_pred_on_test,output_dict=True)        # clasification report from testing data
    df_classification_report_test = pd.DataFrame(testing_report).transpose()               # converting report to dataframe
    auc_score_test=roc_auc_score(y_test, y_pred_prob_test)                                 # auc score on testing data
    auc_report[model_name]= auc_score_test
    precision = precision_score(y_test, y_pred_on_test)
    precision_report[model_name] =precision


    print(f"-----Metrics score of {model_name}----- \n")        #printing model name
    # print(f"best params K :{gs.best_params_}")
    # print(f"errors on training data {n_errors_train} : \n")     # printing errors on training data
    print(f"Accuracy Score on training data : {accuracy_score(y_train, y_pred_on_train)}") # printing accuracy score on train data
    print(f"Confusion Matrix on training data : \n {confusion_matrix(y_train, y_pred_on_train)}") #training data confusion matrix
    print(f"Classification Report on training data :\n {df_classification_report_train}\n")         # classification report on train data


    print("\n")     # next line
    # print(f"errors on testing data {n_errors_test} : \n")   # printing errors on testing data
    print(f"Accuracy Score on test data : {accuracy_score(y_test, y_pred_on_test)}\n")  # printing accuracy score on testing data
    print(f"Confusion Matrix on testing data : \n {confusion_matrix(y_test, y_pred_on_test)}") #training data confusion matrix
    print(f"Classification Report on testing data : \n {df_classification_report_test}\n")   # classification report printing
    print(f"Auc_score on testing data : \n {auc_score_test}")  #auc score printing
    print(f"Precision Score on test data : {precision_score(y_test, y_pred_on_test)}\n") # precision score on test data



best_model_auc = max(auc_report.values())
```

```
best_model = [i for i,j in auc_report.items() if j==best_model_auc]
print(f"best model is :  {best_model} with auc score {best_model_auc}")

best_model_precision = max(precision_report.values())
best_model_prec_name = [i for i,j in precision_report.items() if j==best_model_precision]
print(f"best model_precision is :  {best_model_prec_name} with precision score {best_model_precision}")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1
  y = column_or_1d(y, warn=True)
-----Metrics score of Logistic Regression-----

Accuracy Score on training data : 0.8152537313432836
Confusion Matrix on training data :
 [[13782  3010]
 [ 3179 13529]]
Classification Report on training data :
               precision    recall  f1-score       support
0              0.812570  0.820748  0.816639  16792.000000
1              0.818006  0.809732  0.813848  16708.000000
accuracy       0.815254  0.815254  0.815254      0.815254
macro avg      0.815288  0.815240  0.815243  33500.000000
weighted avg   0.815281  0.815254  0.815247  33500.000000



Accuracy Score on test data : 0.8112121212121212

Confusion Matrix on testing data :
 [[6685 1523]
 [1592 6700]]
Classification Report on testing data :
               precision    recall  f1-score       support
0              0.807660  0.814449  0.811040   8208.000000
1              0.814788  0.808008  0.811384   8292.000000
accuracy       0.811212  0.811212  0.811212      0.811212
macro avg      0.811224  0.811229  0.811212  16500.000000
weighted avg   0.811242  0.811212  0.811213  16500.000000

Auc_score on testing data :
 0.8952409653636423
Precision Score on test data : 0.8147877903441566

-----Metrics score of Decision Tree-----

Accuracy Score on training data : 1.0
Confusion Matrix on training data :
 [[16792     0]
 [    0 16708]]
Classification Report on training data :
               precision  recall  f1-score  support
0                    1.0     1.0       1.0  16792.0
1                    1.0     1.0       1.0  16708.0
accuracy             1.0     1.0       1.0      1.0
macro avg            1.0     1.0       1.0  33500.0
weighted avg         1.0     1.0       1.0  33500.0



Accuracy Score on test data : 0.7220606060606061

Confusion Matrix on testing data :
 [[5919 2289]
 [2297 5995]]
Classification Report on testing data :
```

LSTM RNN

```
# importing libraries

from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
```

```
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Bidirectional
from keras.callbacks import EarlyStopping
from keras import regularizers


## vocabulary size
voc_size=5000


# one hot encoding
onehot_repr=[one_hot(words,voc_size) for words in corpus]
onehot_repr
```

```
[[1335,
  226,
  211,
  1259,
  1624,
  225,
  2139,
  4595,
  2775,
  425,
  4438,
  435,
  435,
  2050,
  4017,
  282,
  225,
  4507,
  1846,
  41,
  1399,
  4741,
  2775,
  3313,
  29,
  2389,
  2990,
  937,
  2794,
  2317,
  2990,
  3879,
  504,
  588,
  1885,
  4372,
  1399,
  1970,
  4618,
  3737,
  3313,
  435,
  435,
  275,
  225,
  366,
  3740,
  16,
  2655,
  870,
  719,
  373,
  4157,
  634,
  4888,
  4242,
  1150,
  2226,
```

```
# embedding representation

sent_length=20
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)

embedded_docs
```

```
array([[4696,  704,  839, ..., 4300, 2710, 1632],
       [2564, 4224, 4158, ...,  876, 3536, 3603],
       [1008, 3547, 1413, ...,   29, 4049, 4352],
       ...,
       [2195,  974, 4111, ..., 2255, 4161, 2677],
       [1983, 2645, 1225, ..., 3533, 2877, 3965],
       [3090, 1707, 2039, ..., 4502, 3452, 2140]], dtype=int32)
```

```
## Creating LSTM model
embedding_vector_features=100
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(LSTM(100))
# model.add(Dense(1,activation='sigmoid'))
# model.add(Dense(1,activation='sigmoid'),kernel_regularizer=regularizers.l2(l=0.1))
model.add(Dense(1, activation='sigmoid',kernel_regularizer=regularizers.l2(0.01)))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_14 (Embedding)    (None, 20, 100)           500000

 lstm_14 (LSTM)              (None, 100)               80400

 dense_11 (Dense)            (None, 1)                 101

=================================================================
Total params: 580501 (2.21 MB)
Trainable params: 580501 (2.21 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

```
X_final=np.array(embedded_docs)
y_final=np.array(y)
```

```
## train test split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size=0.33, random_state=42)
```

```
## Model Training
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=100,batch_size=32,callbacks=[early_stopping])
```

```
Epoch 1/100
1047/1047 [==============================] - 53s 48ms/step - loss: 0.5093 - accuracy: 0.7509 - val_loss: 0.4622 - val_accuracy: 0.7812
Epoch 2/100
1047/1047 [==============================] - 56s 53ms/step - loss: 0.4161 - accuracy: 0.8099 - val_loss: 0.4638 - val_accuracy: 0.7784
Epoch 3/100
1047/1047 [==============================] - 40s 39ms/step - loss: 0.3702 - accuracy: 0.8330 - val_loss: 0.4752 - val_accuracy: 0.7776
Epoch 4/100
1047/1047 [==============================] - 39s 37ms/step - loss: 0.3276 - accuracy: 0.8564 - val_loss: 0.5035 - val_accuracy: 0.7726
Epoch 5/100
1047/1047 [==============================] - 39s 38ms/step - loss: 0.2898 - accuracy: 0.8752 - val_loss: 0.5332 - val_accuracy: 0.7678
Epoch 6/100
1047/1047 [==============================] - 41s 39ms/step - loss: 0.2529 - accuracy: 0.8937 - val_loss: 0.5954 - val_accuracy: 0.7592
Epoch 7/100
1047/1047 [==============================] - 39s 37ms/step - loss: 0.2185 - accuracy: 0.9111 - val_loss: 0.6646 - val_accuracy: 0.7570
Epoch 8/100
1047/1047 [==============================] - 40s 38ms/step - loss: 0.1825 - accuracy: 0.9297 - val_loss: 0.7250 - val_accuracy: 0.7585
Epoch 9/100
1047/1047 [==============================] - 40s 38ms/step - loss: 0.1525 - accuracy: 0.9441 - val_loss: 0.7582 - val_accuracy: 0.7468
Epoch 10/100
1047/1047 [==============================] - 39s 38ms/step - loss: 0.1280 - accuracy: 0.9569 - val_loss: 0.8057 - val_accuracy: 0.7525
Epoch 11/100
1047/1047 [==============================] - 40s 38ms/step - loss: 0.1083 - accuracy: 0.9661 - val_loss: 0.8497 - val_accuracy: 0.7477
<keras.src.callbacks.History at 0x79c2e698d4e0>
```

```
# prediction
y_pred=model.predict(X_test)
```

```
516/516 [==============================] - 5s 9ms/step
```

```
# converting pred value in to 1,0
y_pred=np.where(y_pred>=0.5,1,0)


# confusion matrix
confusion_matrix(y_test,y_pred)
```

```
array([[6253, 1955],
       [2208, 6084]])
```

```
# accuracy score

print(accuracy_score(y_test,y_pred))
```

```
0.7476969696969697
```

```
# classification report
test_report= classification_report(y_test, y_pred)
print(test_report)
```

```
                   precision    recall  f1-score   support

               0       0.74      0.76      0.75      8208
               1       0.76      0.73      0.75      8292

        accuracy                           0.75     16500
       macro avg       0.75      0.75      0.75     16500
    weighted avg       0.75      0.75      0.75     16500
```

Lstm Bidirectional Model

```
## Creating LSTM Bidirectional model
embedding_vector_features=100
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Bidirectional(LSTM(100)))
# model.add(Dense(1,activation='sigmoid'))
model.add(Dense(1, activation='sigmoid',kernel_regularizer=regularizers.l2(0.01)))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_15 (Embedding)    (None, 20, 100)           500000

 bidirectional_2 (Bidirecti  (None, 200)               160800
 onal)

 dense_12 (Dense)            (None, 1)                 201

=================================================================
Total params: 661001 (2.52 MB)
Trainable params: 661001 (2.52 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

```
X_final=np.array(embedded_docs)
y_final=np.array(y)


## train test split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size=0.33, random_state=42)


## Model Training
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=32)
```

```
Epoch 1/10
1047/1047 [==============================] - 69s 61ms/step - loss: 0.5116 - accuracy: 0.7489 - val_loss: 0.4847 - val_accuracy: 0.7728
Epoch 2/10
```

```
1047/1047 [==============================] - 57s 54ms/step - loss: 0.4229 - accuracy: 0.8063 - val_loss: 0.4579 - val_accuracy: 0.7828
Epoch 3/10
1047/1047 [==============================] - 61s 59ms/step - loss: 0.3712 - accuracy: 0.8344 - val_loss: 0.4853 - val_accuracy: 0.7738
Epoch 4/10
1047/1047 [==============================] - 62s 59ms/step - loss: 0.3231 - accuracy: 0.8586 - val_loss: 0.5222 - val_accuracy: 0.7735
Epoch 5/10
1047/1047 [==============================] - 61s 59ms/step - loss: 0.2776 - accuracy: 0.8819 - val_loss: 0.5623 - val_accuracy: 0.7638
Epoch 6/10
1047/1047 [==============================] - 62s 60ms/step - loss: 0.2326 - accuracy: 0.9047 - val_loss: 0.6525 - val_accuracy: 0.7609
Epoch 7/10
1047/1047 [==============================] - 62s 59ms/step - loss: 0.1909 - accuracy: 0.9244 - val_loss: 0.6640 - val_accuracy: 0.7529
Epoch 8/10
1047/1047 [==============================] - 56s 53ms/step - loss: 0.1538 - accuracy: 0.9439 - val_loss: 0.7849 - val_accuracy: 0.7507
Epoch 9/10
1047/1047 [==============================] - 61s 58ms/step - loss: 0.1236 - accuracy: 0.9585 - val_loss: 0.8424 - val_accuracy: 0.7410
Epoch 10/10
1047/1047 [==============================] - 57s 55ms/step - loss: 0.1020 - accuracy: 0.9680 - val_loss: 0.9965 - val_accuracy: 0.7385
<keras.src.callbacks.History at 0x79c302c36200>
```

```
 # prediction
y_pred=model.predict(X_test)
```

```
516/516 [==============================] - 7s 13ms/step
```

```
# converting pred value in to 1,0
y_pred=np.where(y_pred>=0.5,1,0)
```

```
# confusion matrix
confusion_matrix(y_test,y_pred)
```

```
array([[5774, 2434],
       [1880, 6412]])
```

```
# accuracy score

print(accuracy_score(y_test,y_pred))
```

```
0.7385454545454545
```

```
# classification report
test_report= classification_report(y_test, y_pred)
print(test_report)
```

```
                  precision    recall  f1-score   support

               0       0.75      0.70      0.73      8208
               1       0.72      0.77      0.75      8292

        accuracy                           0.74     16500
       macro avg       0.74      0.74      0.74     16500
    weighted avg       0.74      0.74      0.74     16500
```

Word2Vec implementation

```
# !pip install gensim
```

```
from tqdm.notebook import tqdm_notebook
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
import gensim.downloader as api
```

```
# wv = api.load('word2vec-google-news-300')
```

```
# initializing lemmetizing

lemmatizer=WordNetLemmatizer()
```

```
# implementing lemmetization

corpus_lem = []
for i in range(0, len(movie_reviews)):
    review = re.sub('[^a-zA-Z0-9]', ' ', movie_reviews['review'][i])
    review = review.lower()
    review = review.split()
    review = [lemmatizer.lemmatize(word) for word in review if not word in stopwords.words('english')]
    print("no. of iterations :", i)
    review = ' '.join(review)
    corpus_lem.append(review)
```

```
Streaming output truncated to the last 5000 lines.
no. of iterations : 45000
no. of iterations : 45001
no. of iterations : 45002
no. of iterations : 45003
no. of iterations : 45004
no. of iterations : 45005
no. of iterations : 45006
no. of iterations : 45007
no. of iterations : 45008
no. of iterations : 45009
no. of iterations : 45010
no. of iterations : 45011
no. of iterations : 45012
no. of iterations : 45013
no. of iterations : 45014
no. of iterations : 45015
no. of iterations : 45016
no. of iterations : 45017
no. of iterations : 45018
no. of iterations : 45019
no. of iterations : 45020
no. of iterations : 45021
no. of iterations : 45022
no. of iterations : 45023
no. of iterations : 45024
no. of iterations : 45025
no. of iterations : 45026
no. of iterations : 45027
no. of iterations : 45028
no. of iterations : 45029
no. of iterations : 45030
no. of iterations : 45031
no. of iterations : 45032
no. of iterations : 45033
no. of iterations : 45034
no. of iterations : 45035
no. of iterations : 45036
no. of iterations : 45037
no. of iterations : 45038
no. of iterations : 45039
no. of iterations : 45040
no. of iterations : 45041
no. of iterations : 45042
no. of iterations : 45043
no. of iterations : 45044
no. of iterations : 45045
no. of iterations : 45046
no. of iterations : 45047
no. of iterations : 45048
no. of iterations : 45049
no. of iterations : 45050
no. of iterations : 45051
no. of iterations : 45052
no. of iterations : 45053
no. of iterations : 45054
no. of iterations : 45055
no. of iterations : 45056
```

```
corpus_lem
```

```
['one reviewer mentioned watching 1 oz episode hooked right exactly happened br br first thing struck oz brutality unflinching scene violence set right word go trust show faint hearted timid show pull punch regard drug sex violence hardcore classic use word br br called oz nickname given oswald maximum security state penitentary focus mainly emerald city experimental section prison cell glass front face inwards privacy high agenda em city home many aryan muslim gangsta latino christian italian irish scuffle death stare dodgy dealing shady agreement never far away br br would say main appeal show due fact go show dare forget pretty picture painted mainstream audience forget charm forget romance oz mess around first episode ever saw struck nasty surreal say ready watched developed taste oz got accustomed high level graphic violence violence injustice crooked guard sold nickel inmate kill order get away well mannered middle class inmate turned prison bitch due lack street skill prison experience watching oz may become comfortable
```

```
 uncomfortable viewing thats get touch darker side',
  'wonderful little production br br filming technique unassuming old time bbc fashion give comforting sometimes discomforting sense
 realism entire piece br br actor extremely well chosen michael sheen got polari voice pat truly see seamless editing guided reference
 williams diary entry well worth watching terrificly written performed piece masterful production one great master comedy life br br
 realism really come home little thing fantasy guard rather use traditional dream technique remains solid disappears play knowledge
 sens particularly scene concerning orton halliwell set particularly flat halliwell mural decorating every surface terribly well
 done',
  'thought wonderful way spend time hot summer weekend sitting air conditioned theater watching light hearted comedy plot simplistic
 dialogue witty character likable even well bread suspected serial killer may disappointed realize match point 2 risk addiction
 thought proof woody allen still fully control style many u grown love br br laughed one woody comedy year dare say decade never
 impressed scarlet johanson managed tone sexy image jumped right average spirited young woman br br may crown jewel career wittier
 devil wear prada interesting superman great comedy go see friend',
  'basically family little boy jake think zombie closet parent fighting time br br movie slower soap opera suddenly jake decides
 become rambo kill zombie br br ok first going make film must decide thriller drama drama movie watchable parent divorcing arguing
 like real life jake closet totally ruin film expected see boogeyman similar movie instead watched drama meaningless thriller spot br
 br 3 10 well playing parent descent dialog shot jake ignore',
  'petter mattei love time money visually stunning film watch mr mattei offer u vivid portrait human relation movie seems telling u
 money power success people different situation encounter br br variation arthur schnitzler play theme director transfer action
 present time new york different character meet connect one connected one way another next person one seems know previous point
 contact stylishly film sophisticated luxurious look taken see people live world live habitat br br thing one get soul picture
 different stage loneliness one inhabits big city exactly best place human relation find sincere fulfillment one discerns case people
 encounter br br acting good mr mattei direction steve buscemi rosario dawson carol kane michael imperioli adrian grenier rest
 talented cast make character come alive br br wish mr mattei good luck await anxiously next work',
  'probably time favorite movie story selflessness sacrifice dedication noble cause preachy boring never get old despite seen 15 time
 last 25 year paul lukas performance brings tear eye bette davis one truly sympathetic role delight kid grandma say like dressed
 midget child make fun watch mother slow awakening happening world roof believable startling dozen thumb movie',
  'sure would like see resurrection dated seahunt series tech today would bring back kid excitement grew black white tv seahunt
 gunsmoke hero every week vote comeback new sea hunt need change pace tv would work world water adventure oh way thank outlet like
 view many viewpoint tv many movie ole way believe got wanna say would nice read plus point sea hunt rhyme would 10 line would let
 submit leave doubt quit must go let',
  'show amazing fresh innovative idea 70 first aired first 7 8 year brilliant thing dropped 1990 show really funny anymore continued
 decline complete waste time today br br truly disgraceful far show fallen writing painfully bad performance almost bad mildly
 entertaining respite guest host show probably still air find hard believe creator hand selected original cast also chose band hack
 followed one recognize brilliance see fit replace mediocrity felt must give 2 star respect original cast made show huge success show
 awful believe still air',
  'encouraged positive comment film looking forward watching film bad mistake seen 950 film truly one worst awful almost every way
 editing pacing storyline acting soundtrack film song lame country tune played le four time film look cheap nasty boring extreme
 rarely happy see end credit film br br thing prevents giving 1 score harvey keitel far best performance least seems making bit effort
 one keitel obsessive',
  'like original gut wrenching laughter like movie young old love movie hell even mom liked br br great camp',
  'phil alien one quirky film humour based around oddness everything rather actual punchlines br br first odd pretty funny movie
 progressed find joke oddness funny anymore br br low budget film thats never problem pretty interesting character eventually lost
 interest br br imagine film would appeal stoner currently partaking br br something similar better try brother another planet',
  'saw movie 12 came recall scariest scene big bird eating men dangling helplessly parachute right air horror horror br br young kid
 going cheesy b film saturday afternoon still tired formula monster type movie usually included hero beautiful woman might daughter
 professor happy resolution monster died end care much romantic angle 12 year old predictable plot love unintentional humor br br year
 later saw psycho came loved star janet leigh bumped early film sat took notice point since screenwriter making story make scary
 possible well worn formula rule',
  'im big fan boll work many enjoyed movie postal maybe im one boll apparently bought right use far cry long ago even game even
 finsished br br people enjoyed killing mercs infiltrating secret research lab located tropical island warned far cry something mr
```

```
# saving corpus in to dataframe

corpus_lem_data= pd.DataFrame(corpus_lem,columns=['review'])


# adding y column in to corpus_data dataframe

corpus_lem_data['sentiment']=movie_reviews['sentiment']


corpus_lem_data
```

	review	sentiment
0	one reviewer mentioned watching 1 oz episode h	positive

```
corpus_lem_data.to_csv("corpus_lem.txt")
```

```
len(corpus_lem)
```

```
50000
```

```
# simple_preprocess lower each word in sentence

from nltk import sent_tokenize
from gensim.utils import simple_preprocess
```

```
corpus_lem[49999]
```

```
'one expects star trek movie high art fan expect movie good best episode unfortunately movie muddled implausible plot left cringing far worst nine far movie even chance watch well known character interact another movie save movie including goofy scene kirk spock mccoy yo semite br br would say movie worth rental hardly worth watching however true fan need see movie renting movie way see even cable channe l avoid movie'
```

```
# tokenization of sentences in corpus
words=[]
for sent in corpus_lem:
    sent_token=sent_tokenize(sent)
    for sent in sent_token:
        words.append(simple_preprocess(sent))
```

```
words
```

```
[['one',
  'reviewer',
  'mentioned',
  'watching',
  'oz',
  'episode',
  'hooked',
  'right',
  'exactly',
  'happened',
  'br',
  'br',
  'first',
  'thing',
  'struck',
  'oz',
  'brutality',
  'unflinching',
  'scene',
  'violence',
  'set',
  'right',
  'word',
  'go',
  'trust',
  'show',
  'faint',
  'hearted',
  'timid',
  'show',
  'pull',
  'punch',
  'regard',
  'drug',
  'sex',
  'violence',
  'hardcore',
  'classic',
  'use',
  'word',
  'br',
  'br',
  'called',
  'oz',
  'nickname',
  'given',
  'oswald',
  'maximum',
  'security',
```

```
 'state',
 'penitentary',
 'focus',
 'mainly',
 'emerald',
 'city',
 'experimental',
 'section',
 'prison',
```

```
len(words)
```

```
50000
```

```
### Lets train Word2vec from scratch
import gensim
```

```
model=gensim.models.Word2Vec(words,window=5,min_count=2,workers=-1)
```

```
WARNING:gensim.models.word2vec:EPOCH 0: supplied example count (0) did not equal expected count (50000)
WARNING:gensim.models.word2vec:EPOCH 0: supplied raw word count (0) did not equal expected count (6082360)
WARNING:gensim.models.word2vec:EPOCH 1: supplied example count (0) did not equal expected count (50000)
WARNING:gensim.models.word2vec:EPOCH 1: supplied raw word count (0) did not equal expected count (6082360)
WARNING:gensim.models.word2vec:EPOCH 2: supplied example count (0) did not equal expected count (50000)
WARNING:gensim.models.word2vec:EPOCH 2: supplied raw word count (0) did not equal expected count (6082360)
WARNING:gensim.models.word2vec:EPOCH 3: supplied example count (0) did not equal expected count (50000)
WARNING:gensim.models.word2vec:EPOCH 3: supplied raw word count (0) did not equal expected count (6082360)
WARNING:gensim.models.word2vec:EPOCH 4: supplied example count (0) did not equal expected count (50000)
WARNING:gensim.models.word2vec:EPOCH 4: supplied raw word count (0) did not equal expected count (6082360)
```

```
model.wv.index_to_key
```

```
['br',
 'movie',
 'film',
 'one',
 'like',
 'time',
 'good',
 'character',
 'story',
 'even',
 'get',
 'would',
 'make',
 'see',
 'really',
 'scene',
 'well',
 'much',
 'bad',
 'people',
 'great',
 'also',
 'first',
 'show',
 'way',
 'thing',
 'made',
 'life',
 'could',
 'think',
 'go',
 'know',
 'watch',
 'love',
 'plot',
 'actor',
 'two',
 'many',
 'seen',
 'year',
 'say',
 'end',
 'never',
 'acting',
 'look',
 'best',
 'little',
 'ever',
```

```
 'man',
 'better',
 'take',
 'come',
 'work',
 'still',
 'part',
 'something',
 'director',
 'find',
```

```
# corpus count in model

model.corpus_count
```

```
50000
```

```
#corpus total words count

model.corpus_total_words
```

```
6082360
```

```
model.wv.most_similar('happy')
```

```
[('fathered', 0.46972110867500305),
 ('flattest', 0.4232831299304962),
 ('reccomend', 0.41049301624298096),
 ('downhearted', 0.3810591697692871),
 ('titillation', 0.36848196387290955),
 ('rickie', 0.36054477095603943),
 ('triad', 0.35657837986946106),
 ('selznick', 0.3549466133117676),
 ('ajay', 0.353021502494812),
 ('retooled', 0.3513887822628021)]
```

```
model.wv.get_index('realism')
```

```
1637
```