Movielens Case Study

DESCRIPTION

Background of Problem Statement :

The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. Members of the GroupLens Research Project are involved in many research projects related to the fields of information filtering, collaborative filtering, and recommender systems. The project is led by professors John Riedl and Joseph Konstan. The project began to explore automated collaborative filtering in 1992 but is most well known for its worldwide trial of an automated collaborative filtering system for Usenet news in 1996. Since then the project has expanded its scope to research overall information by filtering solutions, integrating into content-based methods, as well as, improving current collaborative filtering technology.

Problem Objective :

Here, we ask you to perform the analysis using the Exploratory Data Analysis technique. You need to find features affecting the ratings of any particular movie and build a model to predict the movie ratings.

Domain: Entertainment

Analysis Tasks to be performed:

1. Import the three datasets
2. Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserId)
3. Explore the datasets using visual representations (graphs or tables), also include your comments on the following: User Age Distribution User rating of the movie “Toy Story” Top 25 movies by viewership rating Find the ratings for all the movies reviewed by for a particular user of user id = 2696
4. Feature Engineering:

```
    Use column genres:

 Find out all the unique genres (Hint: split the data in column genre
making a list and then process the data to find out only the unique
categories of genres)

 Create a separate column for each genre category with a one-hot encoding
( 1 and 0) whether or not the movie belongs to that genre.

 Determine the features affecting the ratings of any particular movie.

 Develop an appropriate model to predict the movie ratings
```

Dataset Description :

These files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000.

Ratings.dat Format - UserID::MovieID::Rating::Timestamp

Field Description UserID Unique identification for each user MovieID Unique identification for each movie Rating User rating for each movie Timestamp Timestamp generated while adding user review UserIDs range between 1 and 6040 The MovieIDs range between 1 and 3952 Ratings are made on a 5-star scale (whole-star ratings only) A timestamp is represented in seconds since the epoch is returned by time(2) Each user has at least 20 ratings

Users.dat Format - UserID::Gender::Age::Occupation::Zip-code

Field Description UserID Unique identification for each user Genere Category of each movie Age User's age Occupation User's Occupation Zip-code Zip Code for the user's location All demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided demographic information are included in this data set.

Gender is denoted by an "M" for male and "F" for female Age is chosen from the following ranges:

Value Description 1 "Under 18" 18 "18-24" 25 "25-34" 35 "35-44" 45 "45-49" 50 "50-55" 56 "56+"

Occupation is chosen from the following choices: Value Description 0 "other" or not specified 1 "academic/educator" 2 "artist" 3 "clerical/admin" 4 "college/grad student" 5 "customer service" 6 "doctor/health care" 7 "executive/managerial" 8 "farmer" 9 "homemaker" 10 "K-12 student" 11 "lawyer" 12 "programmer" 13 "retired" 14 "sales/marketing" 15 "scientist" 16 "self-employed" 17 "technician/engineer" 18 "tradesman/craftsman" 19 "unemployed" 20 "writer"

Movies.dat Format - MovieID::Title::Genres

Field Description MovieID Unique identification for each movie Title A title for each movie Genres Category of each movie

Titles are identical to titles provided by the IMDB (including year of release)

Genres are pipe-separated and are selected from the following genres: Action Adventure Animation Children's Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western Some MovieIDs do not correspond to a movie due to accidental duplicate entries and/or test entries Movies are mostly entered by hand, so errors and inconsistencies may exist

Please download the dataset from here

Good Luck!!

#
Solution

#

In [ ]:
```
# if any error happened during import please run the below commands
#!pip install lightgbm
```

```
#!pip install xgboost
```

1. Import the three datasets

In [ ]:

```
#importing pandas dataframe
import pandas as pd

#importing matplolib
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

#importing reg-ex
import re

#Hold out method for splitting data
from sklearn.model_selection import train_test_split

#importing accuracy_score
from sklearn.metrics import accuracy_score

#importing LGBMClassifier
from lightgbm import LGBMClassifier

#importing xgboost
import xgboost
```

In [ ]:

```
moviesDataFrame = pd.read_csv('Data/movies.dat',sep="::",header=None,names=["MovieID
moviesDataFrame.set_index('MovieID', drop=True)
moviesDataFrame.head(10)
```

Out[ ]:

	MovieID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children's
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller

In [ ]:

```
ratingsDataFrame = pd.read_csv('Data/ratings.dat',sep="::",header=None,names=["UserI
#ratingsDataFrame.set_index('Id', drop=True)
ratingsDataFrame.head(10)
```

Out[ ]:

	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760

	UserID	MovieID	Rating	Timestamp
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291
5	1	1197	3	978302268
6	1	1287	5	978302039
7	1	2804	5	978300719
8	1	594	4	978302268
9	1	919	4	978301368

In [ ]:
```
usersDataFrame = pd.read_csv('Data/users.dat',sep="::",header=None,names=["UserID","
#ratingsDataFrame.set_index('Id', drop=True)
usersDataFrame.head(10)
```

Out[ ]:

	UserID	Gender	Age	Occupation	Zip-code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455
5	6	F	50	9	55117
6	7	M	35	1	06810
7	8	M	25	12	11413
8	9	M	25	17	61614
9	10	F	35	1	95370

1. Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserId)

In [ ]:
```
masterDataFrame = moviesDataFrame.merge(ratingsDataFrame, how="inner", on="MovieID")
masterDataFrame = masterDataFrame.merge(usersDataFrame, how="inner", on="UserID")
masterDataFrame.dropna(inplace=True)
masterDataFrame.head()
```

Out[ ]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	A
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	A
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	

1. Explore the datasets using visual representations (graphs or tables), also include your comments on the following:

   User Age Distribution

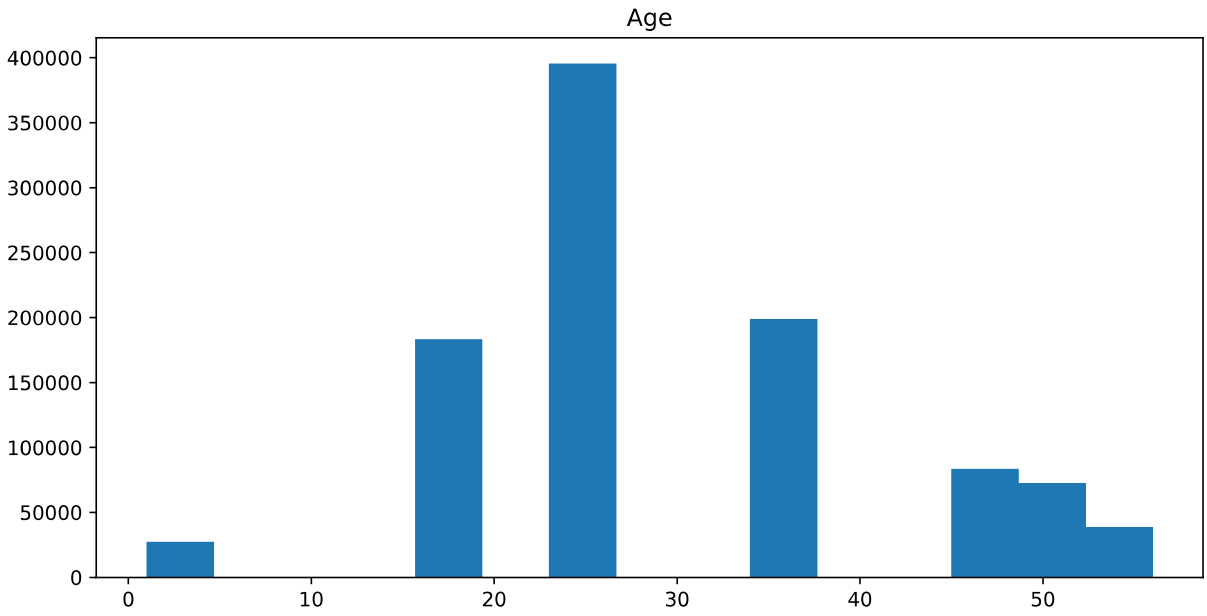
   User rating of the movie “Toy Story”

   Top 25 movies by viewership rating

   Find the ratings for all the movies reviewed by for a particular user of user id = 2696

User Age Distribution

In [ ]:
```
_=masterDataFrame.hist('Age', bins=15,figsize=(10,5),grid=False)
```



User rating of the movie “Toy Story”

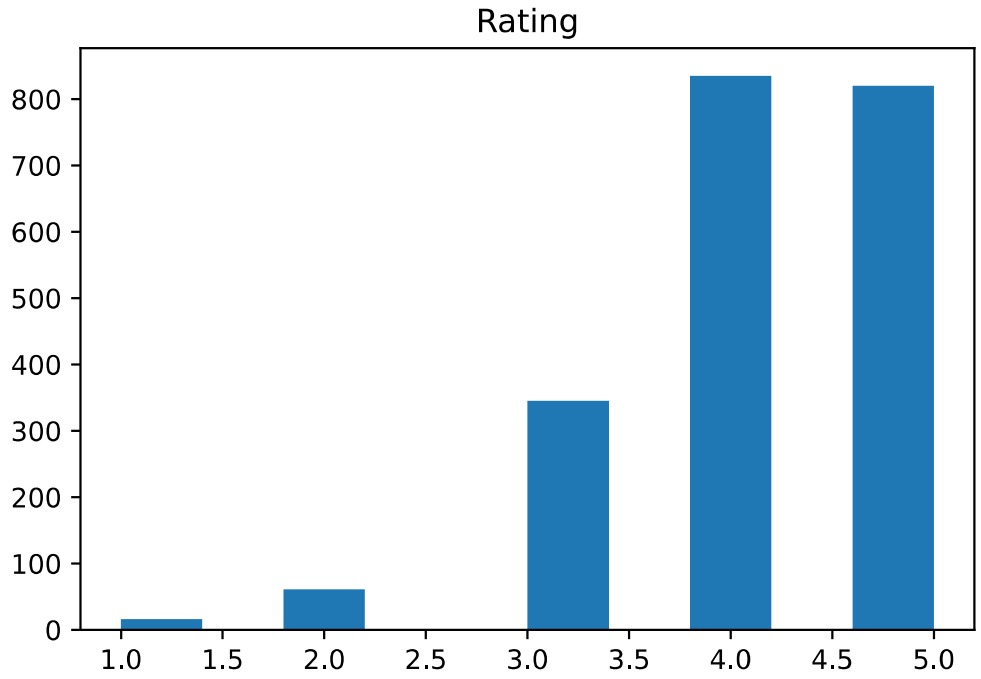
In [ ]:
```
filteredDataFrame = masterDataFrame[masterDataFrame.MovieID.eq(1)]
sortedDataframe= filteredDataFrame.sort_values( ["Title","UserID","Rating"],ascendin
sortedDataframe[["Title","Rating","UserID"]].head(25)
```

Out[ ]:

	Title	Rating	UserID

	Title	Rating	UserID
0	Toy Story (1995)	5	1
53	Toy Story (1995)	4	6
124	Toy Story (1995)	4	8
263	Toy Story (1995)	5	9
369	Toy Story (1995)	5	10
770	Toy Story (1995)	4	18
1075	Toy Story (1995)	5	19
1330	Toy Story (1995)	3	21
1352	Toy Story (1995)	4	23
1656	Toy Story (1995)	3	26
2056	Toy Story (1995)	3	28
2163	Toy Story (1995)	5	34
2327	Toy Story (1995)	5	36
2678	Toy Story (1995)	5	38
2778	Toy Story (1995)	5	44
2971	Toy Story (1995)	4	45
3268	Toy Story (1995)	4	48
3866	Toy Story (1995)	5	49
3974	Toy Story (1995)	5	51
4014	Toy Story (1995)	5	56
4081	Toy Story (1995)	4	60
4151	Toy Story (1995)	5	65
4272	Toy Story (1995)	3	68
4344	Toy Story (1995)	3	73
4599	Toy Story (1995)	5	75

In [ ]:
```
_=filteredDataFrame.hist("Rating", bins=10,grid=False)
```



Top 25 movies by viewership rating

In [ ]:
```
top25MoviesDataFrame= masterDataFrame.groupby(["MovieID","Title"]).Rating.count().so
top25MoviesDataFrame.head(25)
```

Out[ ]:
```
MovieID  Title
2858     American Beauty (1999)                                3428
260      Star Wars: Episode IV - A New Hope (1977)             2991
1196     Star Wars: Episode V - The Empire Strikes Back (1980) 2990
1210     Star Wars: Episode VI - Return of the Jedi (1983)     2883
480      Jurassic Park (1993)                                  2672
2028     Saving Private Ryan (1998)                            2653
589      Terminator 2: Judgment Day (1991)                     2649
2571     Matrix, The (1999)                                    2590
1270     Back to the Future (1985)                             2583
593      Silence of the Lambs, The (1991)                      2578
1580     Men in Black (1997)                                   2538
1198     Raiders of the Lost Ark (1981)                        2514
608      Fargo (1996)                                          2513
2762     Sixth Sense, The (1999)                               2459
110      Braveheart (1995)                                     2443
2396     Shakespeare in Love (1998)                            2369
1197     Princess Bride, The (1987)                            2318
527      Schindler's List (1993)                               2304
1617     L.A. Confidential (1997)                              2288
1265     Groundhog Day (1993)                                  2278
1097     E.T. the Extra-Terrestrial (1982)                     2269
2628     Star Wars: Episode I - The Phantom Menace (1999)      2250
2997     Being John Malkovich (1999)                           2241
318      Shawshank Redemption, The (1994)                      2227
858      Godfather, The (1972)                                 2223
Name: Rating, dtype: int64
```

Find the ratings for all the movies reviewed by for a particular user of user id = 2696

In [ ]:
```
userMoviesDataFrame = masterDataFrame[masterDataFrame["UserID"].eq(2696)]
userMoviesDataFrame
```

Out[ ]:

MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Ag

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Ag
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	2
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	2
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	2
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	2
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M	2
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M	2
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M	2
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	2
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M	2
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	973308920	M	2
991045	1645	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller	2696	4	973308904	M	2
991046	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery	2696	4	973308904	M	2
991047	1783	Palmetto (1998)	Film-Noir|Mystery|Thriller	2696	4	973308865	M	2
991048	1805	Wild Things (1998)	Crime|Drama|Mystery|Thriller	2696	4	973308886	M	2
991049	1892	Perfect Murder, A (1998)	Mystery|Thriller	2696	4	973308904	M	2
991050	2338	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller	2696	2	973308920	M	2

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Ag
991051	2389	Psycho (1998)	Crime|Horror|Thriller	2696	4	973308710	M	2
991052	2713	Lake Placid (1999)	Horror|Thriller	2696	1	973308710	M	2
991053	3176	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller	2696	4	973308865	M	2
991054	3386	JFK (1991)	Drama|Mystery	2696	1	973308842	M	2

1. Feature Engineering: Use column genres

Find out all the unique genres (Hint: split the data in column genre making a list and then process the data to find out only the unique categories of genres)

In [ ]:
```
GenereList = masterDataFrame.Genres.tolist()
filteredList = []
i = 0
while(i<len(GenereList)):
    filteredList+= GenereList[i].split('|')
    i+=1

uniqueList = list(set(filteredList))
uniqueList
```

Out[ ]:
```
['Western',
 'Fantasy',
 'Action',
 'Musical',
 'Sci-Fi',
 'War',
 'Romance',
 'Thriller',
 'Horror',
 'Comedy',
 "Children's",
 'Drama',
 'Mystery',
 'Documentary',
 'Crime',
 'Animation',
 'Adventure',
 'Film-Noir']
```

Create a separate column for each genre category with a one-hot encoding (1 and 0) whether or not the movie belongs to that genre.

In [ ]:
```
masterDataFrameClone = masterDataFrame.copy()
masterDataFrameClone
```

Out[ ]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Ge
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	

	MovieID	Title	Genres	UserID	Rating	Timestamp	Ge
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	
...	...	...	...	...	...	...	
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	958489970	
1000205	3535	American Psycho (2000)	Comedy|Horror|Thriller	5727	2	958489970	
1000206	3536	Keeping the Faith (2000)	Comedy|Romance	5727	5	958489902	
1000207	3555	U-571 (2000)	Action|Thriller	5727	3	958490699	
1000208	3578	Gladiator (2000)	Action|Drama	5727	5	958490171	

1000209 rows × 10 columns

In [ ]:
```
masterDataFrameClone=pd.concat([masterDataFrameClone,masterDataFrameClone.Genres.str
masterDataFrameClone
```

Out[ ]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Ge
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	
...	...	...	...	...	...	...	

	MovieID	Title	Genres	UserID	Rating	Timestamp	Ge
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	958489970	
1000205	3535	American Psycho (2000)	Comedy|Horror|Thriller	5727	2	958489970	
1000206	3536	Keeping the Faith (2000)	Comedy|Romance	5727	5	958489902	
1000207	3555	U-571 (2000)	Action|Thriller	5727	3	958490699	
1000208	3578	Gladiator (2000)	Action|Drama	5727	5	958490171	

1000209 rows × 28 columns

Determine the features affecting the ratings of any particular movie.

In [ ]:
```
x = masterDataFrameClone.drop(['UserID','MovieID','Rating','Title', 'Genres', 'Gende
x.shape
```

Out[ ]:
```
(1000209, 21)
```

In [ ]:
```
y = masterDataFrameClone.Rating
y.shape
```

Out[ ]:
```
(1000209,)
```

In [ ]:
```
x.Occupation.value_counts()
```

Out[ ]:
```
4     131032
0     130499
7     105425
1      85351
17     72816
20     60397
12     57214
2      50068
14     49109
16     46021
6      37205
3      31623
10     23290
15     22951
5      21850
11     20563
19     14904
13     13754
18     12086
9      11345
8       2706
Name: Occupation, dtype: int64
```

In [ ]:
```
x = x.join(pd.get_dummies(x.Occupation,prefix='Occupation'))
```

```
x.head(),x.columns
```

Out[ ]:
```
(   Timestamp  Age  Occupation  Action  Adventure  Animation  Children's  \
 0  978824268    1          10       0          0          1           1
 1  978824351    1          10       0          0          1           1
 2  978301777    1          10       0          0          0           0
 3  978300760    1          10       1          1          0           0
 4  978824195    1          10       0          0          0           0

    Comedy  Crime  Documentary  ...  Occupation_11  Occupation_12  \
 0       1      0            0  ...              0              0
 1       0      0            0  ...              0              0
 2       0      0            0  ...              0              0
 3       0      0            0  ...              0              0
 4       0      0            0  ...              0              0

    Occupation_13  Occupation_14  Occupation_15  Occupation_16  Occupation_17  \
 0              0              0              0              0              0
 1              0              0              0              0              0
 2              0              0              0              0              0
 3              0              0              0              0              0
 4              0              0              0              0              0

    Occupation_18  Occupation_19  Occupation_20
 0              0              0              0
 1              0              0              0
 2              0              0              0
 3              0              0              0
 4              0              0              0

 [5 rows x 42 columns],
 Index(['Timestamp', 'Age', 'Occupation', 'Action', 'Adventure', 'Animation',
        'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
        'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',
        'Thriller', 'War', 'Western', 'Occupation_0', 'Occupation_1',
        'Occupation_2', 'Occupation_3', 'Occupation_4', 'Occupation_5',
        'Occupation_6', 'Occupation_7', 'Occupation_8', 'Occupation_9',
        'Occupation_10', 'Occupation_11', 'Occupation_12', 'Occupation_13',
        'Occupation_14', 'Occupation_15', 'Occupation_16', 'Occupation_17',
        'Occupation_18', 'Occupation_19', 'Occupation_20'],
       dtype='object'))
```

Develop an appropriate model to predict the movie ratings

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state =
```

In [ ]:
```
lgb = LGBMClassifier(boosting_type = 'gbdt',n_jobs= -1,objective='multiclass')
```

In [ ]:
```
lgb.fit(x_train,y_train)
```

Out[ ]:
```
LGBMClassifier(objective='multiclass')
```

In [ ]:
```
y_pred = lgb.predict(x_test)
```

In [ ]:
```
print('LGBM accuracy score is : ', accuracy_score(y_test,y_pred)*100)
```

```
LGBM accuracy score is :  37.37865048339849
```

In [ ]:
```
xgb = xgboost.XGBClassifier(n_jobs=-1)
```

In [ ]:
```
xgb.fit(x_train,y_train)
```

```
[15:41:32] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[ ]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [ ]:
```
y_pred_xgb = xgb.predict(x_test)
```

In [ ]:
```
print('XGB accuracy score is : ', accuracy_score(y_test,y_pred_xgb )*100)
```

```
XGB accuracy score is :  38.45892362603853
```