Movielens Solution

March 28, 2021

1 *Movielens Case Study*

2 *Import the three datasets*

[1]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Importing the datasets
ratings_header = "UserID::MovieID::Rating::Timestamp".split("::")
users_header = "UserID::Gender::Age::Occupation::Zip-code".split("::")
movies_header = "MovieID::Title::Genres".split("::")
```

[3]:
```
movies = pd.read_csv("movies.dat", sep="::", names= movies_header)
users = pd.read_csv("users.dat", sep="::", names= users_header)
ratings = pd.read_csv("ratings.dat", sep="::", names= ratings_header)
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:1: ParserWarning:
Falling back to the 'python' engine because the 'c' engine does not support
regex separators (separators > 1 char and different from '\s+' are interpreted
as regex); you can avoid this warning by specifying engine='python'.
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:2: ParserWarning:
Falling back to the 'python' engine because the 'c' engine does not support
regex separators (separators > 1 char and different from '\s+' are interpreted
as regex); you can avoid this warning by specifying engine='python'.

/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:3: ParserWarning:
Falling back to the 'python' engine because the 'c' engine does not support
regex separators (separators > 1 char and different from '\s+' are interpreted
as regex); you can avoid this warning by specifying engine='python'.
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[4]:
```
movies.head()
```

[4]:

	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]:
```
ratings.head()
```

[5]:

	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

[6]:
```
users.head()
```

[6]:

	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

3 *Create a new dataset [Master_Data]*

[7]:
```
# Merging the datsets using 'MovieID'
dfMovieRatings = movies.merge(ratings,on='MovieID')
dfMovieRatings.head()
```

[7]:

	MovieID	Title	Genres	UserID	Rating	Timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474

[8]:
```
dfMovieRatings.shape
```

[8]:
```
(1000209, 6)
```

[9]:
```
# Master dataset creation - Merging Movie ratings with users using 'UserID'
Master_data = dfMovieRatings.merge(users,on="UserID")
Master_data.head()
```

[9]:
```
   MovieID                                    Title  \
0        1                         Toy Story (1995)
1       48                        Pocahontas (1995)
2      150                         Apollo 13 (1995)
3      260  Star Wars: Episode IV - A New Hope (1977)
4      527                  Schindler's List (1993)

                                Genres  UserID  Rating  Timestamp Gender  \
0           Animation|Children's|Comedy       1       5  978824268      F
1  Animation|Children's|Musical|Romance       1       5  978824351      F
2                                 Drama       1       5  978301777      F
3       Action|Adventure|Fantasy|Sci-Fi       1       4  978300760      F
4                           Drama|War       1       5  978824195      F

   Age  Occupation Zip-code
0    1          10    48067
1    1          10    48067
2    1          10    48067
3    1          10    48067
4    1          10    48067
```

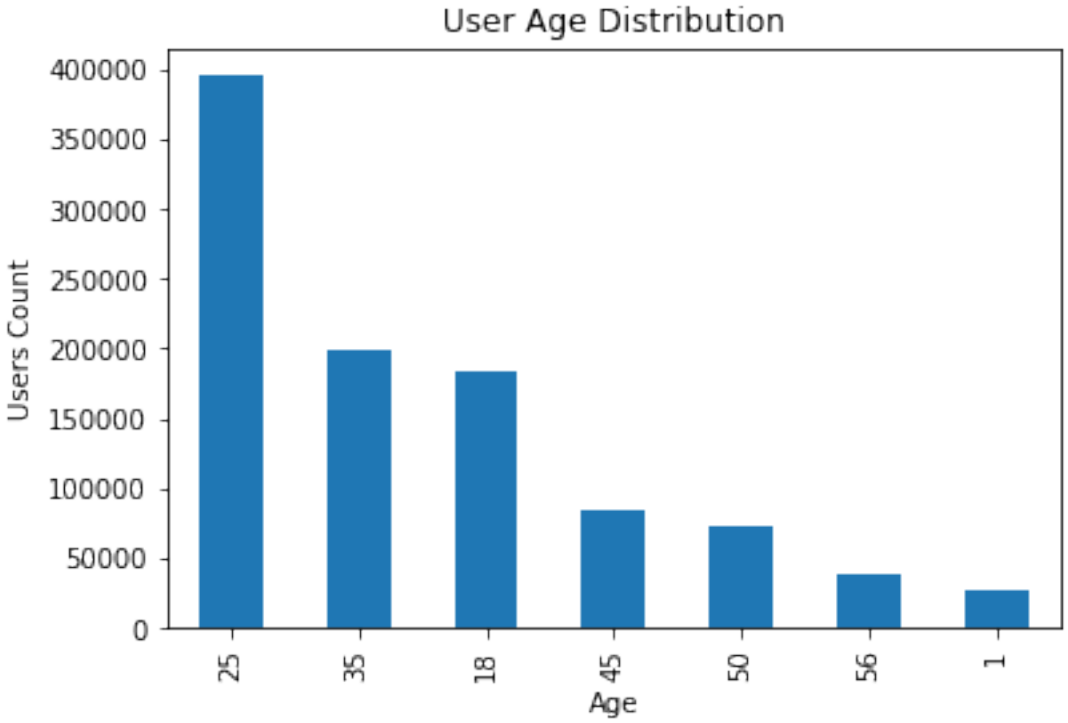
[10]:
```
# Creating a master csv file (Just to save the changes made)
Master_data.to_csv("Master.csv")
```

4 *Data Exploration*

[11]:
```
# 1.User Age Distibution
Master_data['Age'].value_counts()
```

[11]:
```
25    395556
35    199003
18    183536
45     83633
50     72490
56     38780
1      27211
Name: Age, dtype: int64
```

[12]:
```
# Plot to show the user age distribution
Master_data['Age'].value_counts().plot(kind='bar')
plt.xlabel("Age")
plt.title("User Age Distribution")
plt.ylabel('Users Count')
plt.show()
```



[13]:
```
# 2.User Rating for the movie "Toy Story"
toystoryRating = Master_data[Master_data['Title'].str.contains('Toy Story') ==␣
↪True]
toystoryRating
```

[13]:

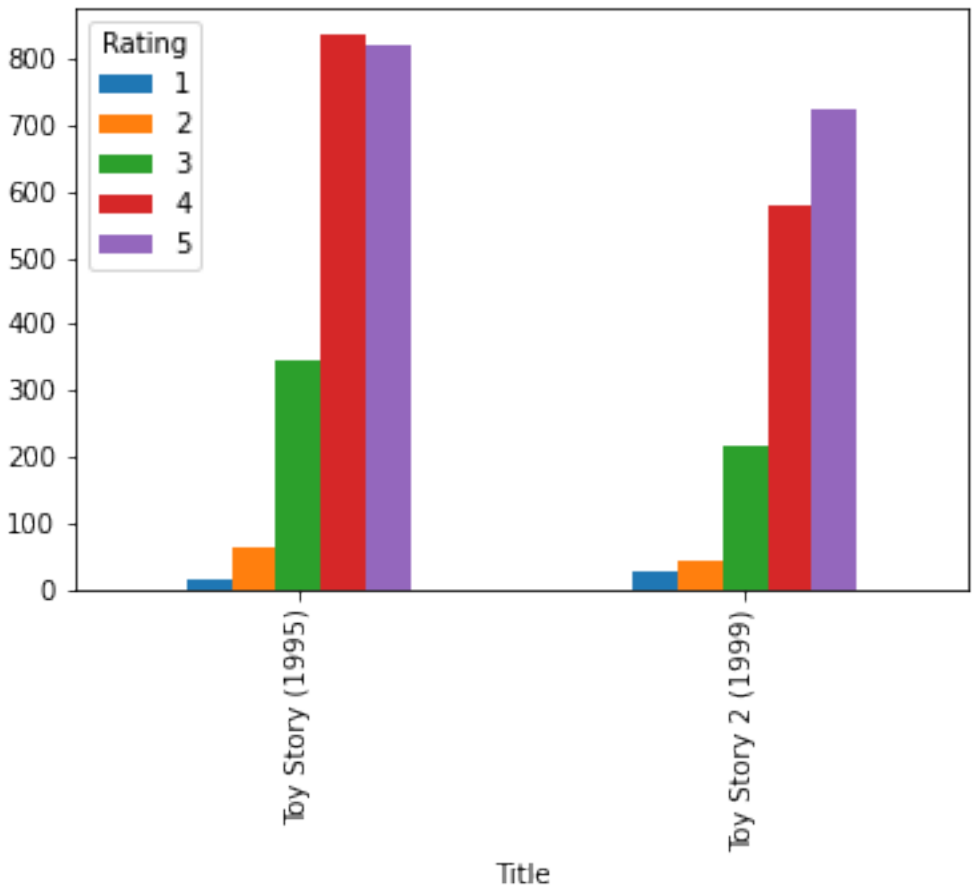
	MovieID	Title	Genres	UserID	\
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	
50	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	1	
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	
…	…	…	…	…	
998988	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	3023	
999027	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	5800	
999486	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	2189	
999869	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	159	

```
1000192     3114  Toy Story 2 (1999)  Animation|Children's|Comedy    5727

         Rating  Timestamp Gender  Age  Occupation Zip-code
0             5  978824268      F    1          10    48067
50            4  978302174      F    1          10    48067
53            4  978237008      F   50           9    55117
124           4  978233496      M   25          12    11413
263           5  978225952      M   25          17    61614
…           …         …    …   …          …       …
998988        4  970471948      F   25           7    92108
999027        5  958015250      M   35          18    90804
999486        4  974607816      M    1          10    60148
999869        4  989966944      F   45           0    37922
1000192       5  958492554      M   25           4    92843

[3662 rows x 10 columns]
```

[14]:
```
toystoryRating.groupby(["Title","Rating"]).size()
```

[14]:
```
Title                Rating
Toy Story (1995)     1          16
                     2          61
                     3         345
                     4         835
                     5         820
Toy Story 2 (1999)   1          25
                     2          44
                     3         214
                     4         578
                     5         724
dtype: int64
```

[15]:
```
# Plot to show the Toystory rating
toystoryRating.groupby(["Title","Rating"]).size().unstack().
 ↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```

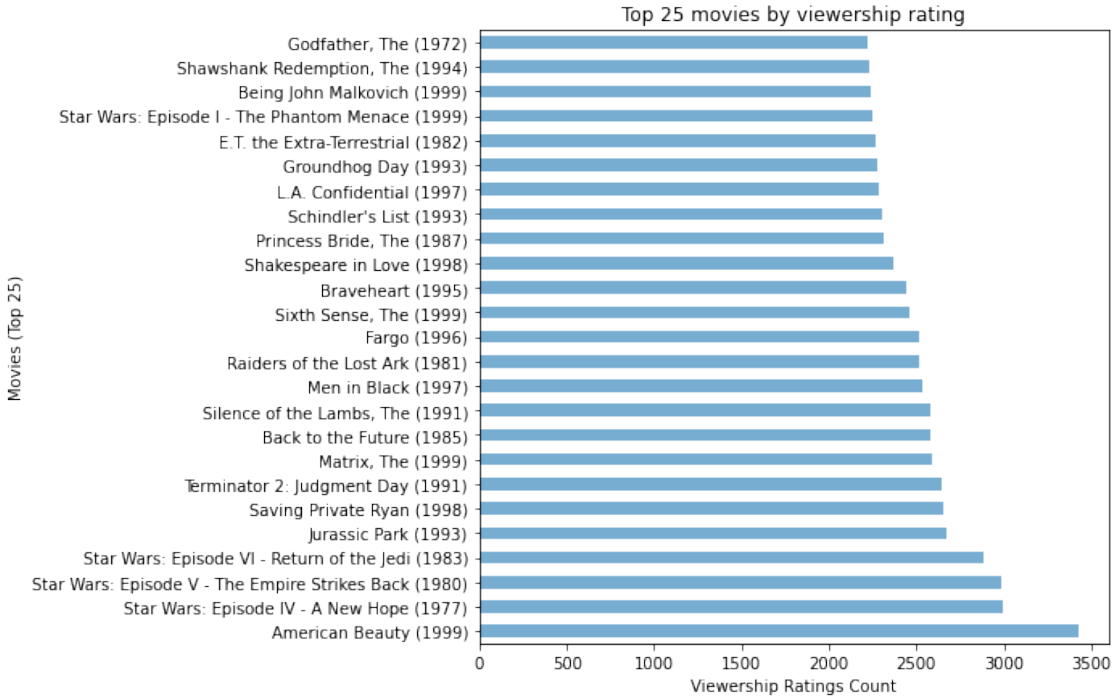


[16]:
```
# 3.Top 25 movies by viewrship ratings
dfTop25 = Master_data.groupby('Title').size().sort_values(ascending=False)[:25]
dfTop25
```

[16]:
```
Title
American Beauty (1999)                                3428
Star Wars: Episode IV - A New Hope (1977)             2991
Star Wars: Episode V - The Empire Strikes Back (1980) 2990
Star Wars: Episode VI - Return of the Jedi (1983)     2883
Jurassic Park (1993)                                  2672
Saving Private Ryan (1998)                            2653
Terminator 2: Judgment Day (1991)                     2649
Matrix, The (1999)                                    2590
Back to the Future (1985)                             2583
Silence of the Lambs, The (1991)                      2578
Men in Black (1997)                                   2538
Raiders of the Lost Ark (1981)                        2514
Fargo (1996)                                          2513
Sixth Sense, The (1999)                               2459
```

```
Braveheart (1995)                                          2443
Shakespeare in Love (1998)                                 2369
Princess Bride, The (1987)                                 2318
Schindler's List (1993)                                    2304
L.A. Confidential (1997)                                   2288
Groundhog Day (1993)                                       2278
E.T. the Extra-Terrestrial (1982)                          2269
Star Wars: Episode I - The Phantom Menace (1999)           2250
Being John Malkovich (1999)                                2241
Shawshank Redemption, The (1994)                           2227
Godfather, The (1972)                                      2223
dtype: int64
```

[17]:
```
# Plot to show the top 25 by viewership ratings
dfTop25.plot(kind='barh',alpha=0.6,figsize=(7,7))
plt.xlabel("Viewership Ratings Count")
plt.ylabel("Movies (Top 25)")
plt.title("Top 25 movies by viewership rating")
plt.show()
```



[18]:
```
# 4.Find the ratings for all the movies reviewed by for a particular user of␣
 ↪user id = 2696
userId = 2696
userRatingById = Master_data[Master_data["UserID"] == userId]
```

```
userRatingById
```

[18]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M			
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M			
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M			
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M			
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M			
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M			
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M			
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M			
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M			
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	973308920	M			
991045	1645	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller	2696	4	973308904	M			
991046	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery	2696	4	973308904	M			
991047	1783	Palmetto (1998)	Film-Noir|Mystery|Thriller	2696	4	973308865	M			
991048	1805	Wild Things (1998)	Crime|Drama|Mystery|Thriller	2696	4	973308886	M			
991049	1892	Perfect Murder, A (1998)	Mystery|Thriller	2696	4	973308904	M			
991050	2338	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller	2696	2	973308920	M			
991051	2389	Psycho (1998)	Crime|Horror|Thriller	2696	4	973308710	M			
991052	2713	Lake Placid (1999)	Horror|Thriller	2696	1	973308710	M			
991053	3176	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller	2696	4	973308865	M			
991054	3386	JFK (1991)	Drama|Mystery	2696	1	973308842	M			

```
991035   25           7    24210
991036   25           7    24210
991037   25           7    24210
991038   25           7    24210
991039   25           7    24210
991040   25           7    24210
991041   25           7    24210
991042   25           7    24210
991043   25           7    24210
991044   25           7    24210
991045   25           7    24210
991046   25           7    24210
991047   25           7    24210
991048   25           7    24210
991049   25           7    24210
991050   25           7    24210
991051   25           7    24210
991052   25           7    24210
991053   25           7    24210
991054   25           7    24210
```

5 *Feature Engineering*

[19]:
```
# Find out all unique genres
# dfGenres = Master_data[]
dfGenres = Master_data['Genres'].str.split("|")
dfGenres
```

[19]:
```
0                    [Animation, Children's, Comedy]
1          [Animation, Children's, Musical, Romance]
2                                            [Drama]
3                [Action, Adventure, Fantasy, Sci-Fi]
4                                         [Drama, War]
                              …
1000204                            [Drama, Thriller]
1000205                   [Comedy, Horror, Thriller]
1000206                            [Comedy, Romance]
1000207                           [Action, Thriller]
1000208                              [Action, Drama]
Name: Genres, Length: 1000209, dtype: object
```

[20]:
```
# 1.To extract unique genres
listGenres = set()
for genre in dfGenres:
    listGenres = listGenres.union(set(genre))
```

```
listGenres
```

[20]:
```
{'Action',
 'Adventure',
 'Animation',
 "Children's",
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Fantasy',
 'Film-Noir',
 'Horror',
 'Musical',
 'Mystery',
 'Romance',
 'Sci-Fi',
 'Thriller',
 'War',
 'Western'}
```

[21]:
```
# 2.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
ratingsOneHot = Master_data['Genres'].str.get_dummies("|")
ratingsOneHot.head()
```

[21]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	\
0	0	0	1	1	1	0	0	
1	0	0	1	1	0	0	0	
2	0	0	0	0	0	0	0	
3	1	1	0	0	0	0	0	
4	0	0	0	0	0	0	0	

	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	\
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	1	0	1	0	
2	1	0	0	0	0	0	0	0	
3	0	1	0	0	0	0	0	1	
4	1	0	0	0	0	0	0	0	

	Thriller	War	Western
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	1	0

[22]:
```
Master_data = pd.concat([Master_data,ratingsOneHot],axis=1)
Master_data.head()
```

[22]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	…	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067	…	0	0	0	0	0	0	0	0	0	0
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067	…	0	0	0	1	0	1	0	0	0	0
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067	…	0	0	0	0	0	0	0	0	0	0
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067	…	1	0	0	0	0	0	1	0	0	0
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067	…	0	0	0	0	0	0	0	0	1	0

[5 rows x 28 columns]

[23]:
```
Master_data.columns
```

[23]:
```
Index(['MovieID', 'Title', 'Genres', 'UserID', 'Rating', 'Timestamp', 'Gender',
       'Age', 'Occupation', 'Zip-code', 'Action', 'Adventure', 'Animation',
       'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
       'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',
       'Thriller', 'War', 'Western'],
      dtype='object')
```

[24]:
```
# Creating a final master file with all operations perfomed till now (Just to␣
 ↪save the changes made)
Master_data.to_csv("Final_Master.csv")
```

[25]:
```
# To determine the features affecting the ratings of the movie
Master_data[["title","Year"]] = Master_data.Title.str.extract("(.)\s\((.
 ↪\d+)",expand=True)
Master_data = Master_data.drop(columns=["title"])
Master_data.head()
```

[25]:

```
   MovieID                                     Title  \
0        1                         Toy Story (1995)
1       48                        Pocahontas (1995)
2      150                         Apollo 13 (1995)
3      260  Star Wars: Episode IV - A New Hope (1977)
4      527                  Schindler's List (1993)

                                Genres  UserID  Rating  Timestamp Gender  \
0           Animation|Children's|Comedy       1       5  978824268      F
1  Animation|Children's|Musical|Romance       1       5  978824351      F
2                               Drama       1       5  978301777      F
3       Action|Adventure|Fantasy|Sci-Fi       1       4  978300760      F
4                          Drama|War       1       5  978824195      F

   Age  Occupation Zip-code  …  Film-Noir  Horror  Musical  Mystery  \
0    1          10    48067  …          0       0        0        0
1    1          10    48067  …          0       0        1        0
2    1          10    48067  …          0       0        0        0
3    1          10    48067  …          0       0        0        0
4    1          10    48067  …          0       0        0        0

   Romance  Sci-Fi  Thriller  War  Western  Year
0        0       0         0    0        0  1995
1        1       0         0    0        0  1995
2        0       0         0    0        0  1995
3        0       1         0    0        0  1977
4        0       0         0    1        0  1993

[5 rows x 29 columns]
```

[26]:
```
Master_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 29 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   MovieID      1000209 non-null  int64
 1   Title        1000209 non-null  object
 2   Genres       1000209 non-null  object
 3   UserID       1000209 non-null  int64
```

```
 4   Rating       1000209 non-null  int64
 5   Timestamp    1000209 non-null  int64
 6   Gender       1000209 non-null  object
 7   Age          1000209 non-null  int64
 8   Occupation   1000209 non-null  int64
 9   Zip-code     1000209 non-null  object
 10  Action       1000209 non-null  int64
 11  Adventure    1000209 non-null  int64
 12  Animation    1000209 non-null  int64
 13  Children's   1000209 non-null  int64
 14  Comedy       1000209 non-null  int64
 15  Crime        1000209 non-null  int64
 16  Documentary  1000209 non-null  int64
 17  Drama        1000209 non-null  int64
 18  Fantasy      1000209 non-null  int64
 19  Film-Noir    1000209 non-null  int64
 20  Horror       1000209 non-null  int64
 21  Musical      1000209 non-null  int64
 22  Mystery      1000209 non-null  int64
 23  Romance      1000209 non-null  int64
 24  Sci-Fi       1000209 non-null  int64
 25  Thriller     1000209 non-null  int64
 26  War          1000209 non-null  int64
 27  Western      1000209 non-null  int64
 28  Year         1000209 non-null  object
dtypes: int64(24), object(5)
memory usage: 228.9+ MB
```

[27]:
```
Master_data['Year'] = Master_data.Year.astype(int)
Master_data['Movie_Age'] = 2000 - Master_data.Year
Master_data.head()
```

[27]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	…	Horror	Musical	Mystery	Romance	Sci-Fi	\
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										
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										

```
0    1          10    48067  …       0        0        0        0       0
1    1          10    48067  …       0        1        0        1       0
2    1          10    48067  …       0        0        0        0       0
3    1          10    48067  …       0        0        0        0       1
4    1          10    48067  …       0        0        0        0       0

   Thriller  War  Western  Year  Movie_Age
0         0    0        0  1995          5
1         0    0        0  1995          5
2         0    0        0  1995          5
3         0    0        0  1977         23
4         0    1        0  1993          7

[5 rows x 30 columns]
```

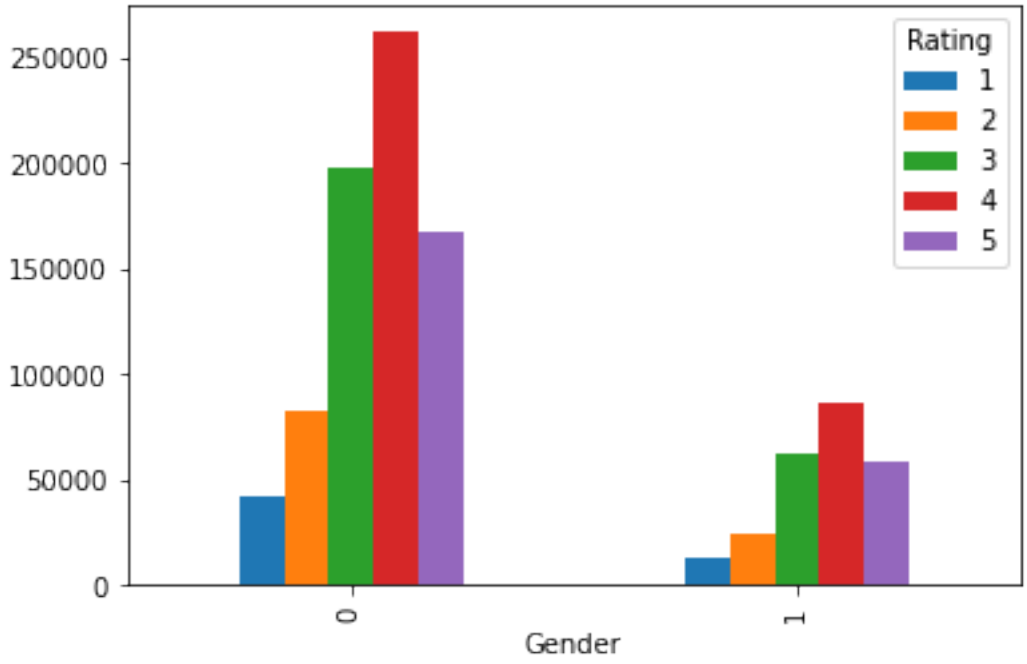
[28]:
```
Master_data['Gender'] = Master_data.Gender.str.replace('F','1')
Master_data['Gender'] = Master_data.Gender.str.replace('M','0')
Master_data['Gender'] = Master_data.Gender.astype(int)
Master_data.head()
```

[28]:
```
   MovieID                                     Title  \
0        1                          Toy Story (1995)
1       48                         Pocahontas (1995)
2      150                          Apollo 13 (1995)
3      260  Star Wars: Episode IV - A New Hope (1977)
4      527                    Schindler's List (1993)

                                 Genres  UserID  Rating  Timestamp  Gender  \
0           Animation|Children's|Comedy       1       5  978824268       1
1  Animation|Children's|Musical|Romance       1       5  978824351       1
2                                 Drama       1       5  978301777       1
3       Action|Adventure|Fantasy|Sci-Fi       1       4  978300760       1
4                             Drama|War       1       5  978824195       1

   Age  Occupation Zip-code  …  Horror  Musical  Mystery  Romance  Sci-Fi  \
0    1          10    48067  …       0        0        0        0       0
1    1          10    48067  …       0        1        0        1       0
2    1          10    48067  …       0        0        0        0       0
3    1          10    48067  …       0        0        0        0       1
4    1          10    48067  …       0        0        0        0       0

   Thriller  War  Western  Year  Movie_Age
0         0    0        0  1995          5
1         0    0        0  1995          5
2         0    0        0  1995          5
3         0    0        0  1977         23
4         0    1        0  1993          7
```

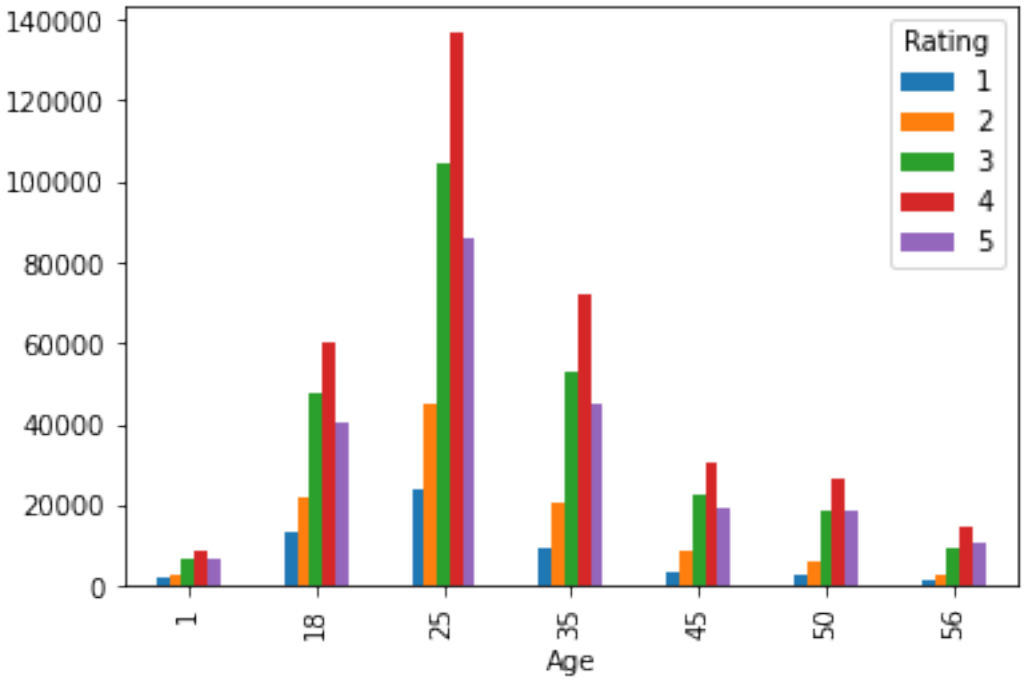
```
[5 rows x 30 columns]
```

[29]:
```
dfGenderAffecting = Master_data.groupby('Gender').size().
 ↪sort_values(ascending=False)[:25]
```

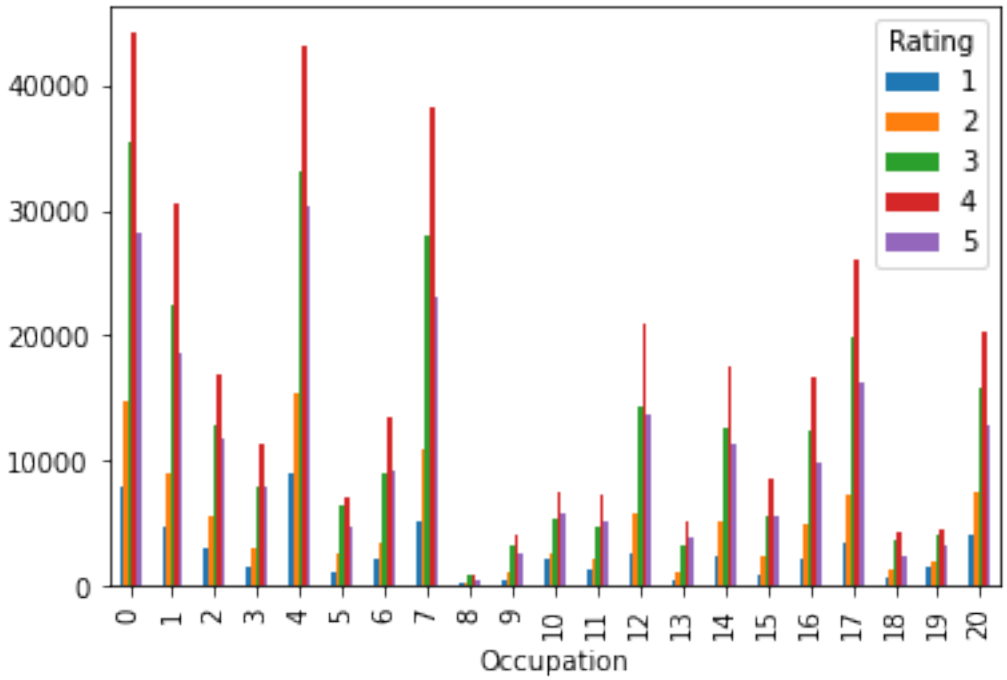
[30]:
```
Master_data.groupby(["Gender","Rating"]).size().unstack().
 ↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```



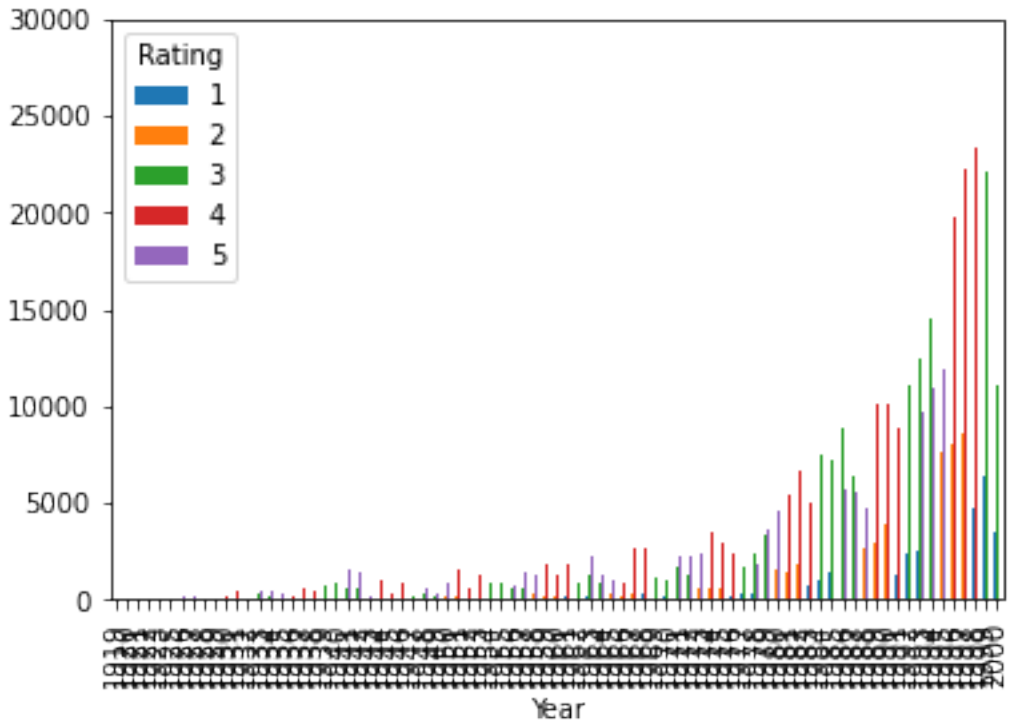
[31]:
```
Master_data.groupby(["Age","Rating"]).size().unstack().
 ↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```



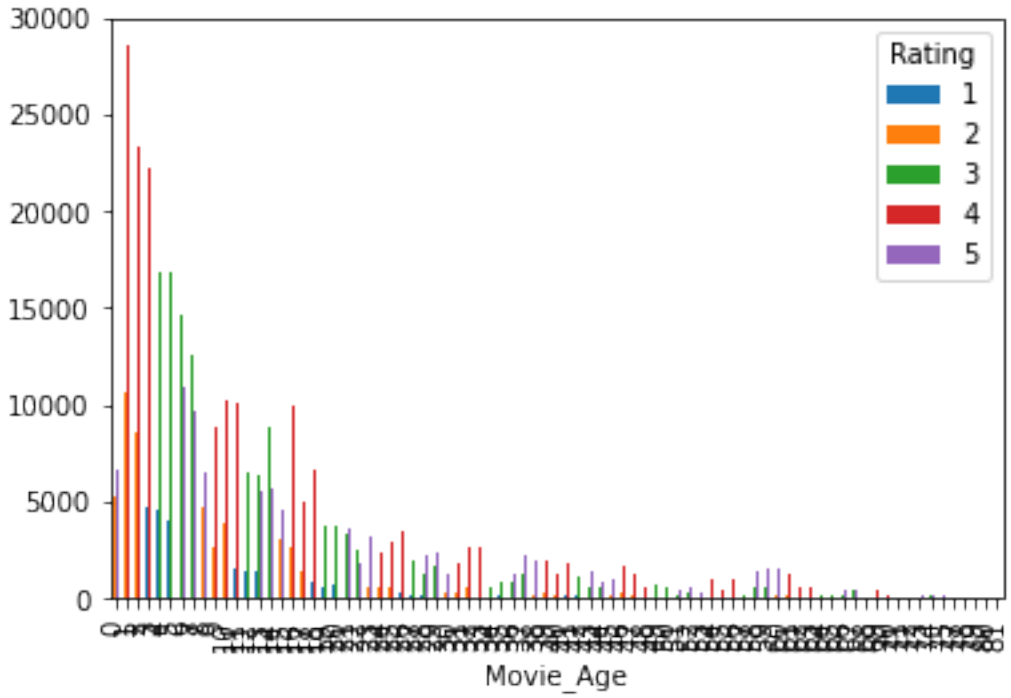
[32]:
```
Master_data.groupby(["Occupation","Rating"]).size().unstack().
 ↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```



[33]:
```
Master_data.groupby(["Year","Rating"]).size().unstack().
 ↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```



[34]:
```
Master_data.groupby(["Movie_Age","Rating"]).size().unstack().
 ↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```



6 *Develop a appropriate model to predict the movie ratings*

[35]:
```
# First 500 extracted records (A Sample)
first_500 = Master_data[:1000]
first_500
```

[35]:

	MovieID	Title	\
0	1	Toy Story (1995)	
1	48	Pocahontas (1995)	
2	150	Apollo 13 (1995)	
3	260	Star Wars: Episode IV - A New Hope (1977)	
4	527	Schindler's List (1993)	
..	…	…	
995	2384	Babe: Pig in the City (1998)	
996	2391	Simple Plan, A (1998)	
997	2394	Prince of Egypt, The (1998)	
998	2402	Rambo: First Blood Part II (1985)	
999	2404	Rambo III (1988)	

	Genres	UserID	Rating	Timestamp	Gender	\
0	Animation|Children's|Comedy	1	5	978824268	1	
1	Animation|Children's|Musical|Romance	1	5	978824351	1	

```
2                                    Drama       1       5  978301777       1
3         Action|Adventure|Fantasy|Sci-Fi       1       4  978300760       1
4                             Drama|War       1       5  978824195       1
..                                      …     …      …       …     …
995                   Children's|Comedy      18       2  978155233       1
996                      Crime|Thriller      18       1  978155685       1
997                   Animation|Musical      18       4  978154907       1
998                          Action|War      18       2  978153894       1
999                          Action|War      18       2  978153977       1

     Age  Occupation Zip-code  …  Horror  Musical  Mystery  Romance  Sci-Fi  \
0      1          10    48067  …       0        0        0        0       0
1      1          10    48067  …       0        1        0        1       0
2      1          10    48067  …       0        0        0        0       0
3      1          10    48067  …       0        0        0        0       1
4      1          10    48067  …       0        0        0        0       0
..   …          …       …  …     …       …        …      …       …
995   18           3    95825  …       0        0        0        0       0
996   18           3    95825  …       0        0        0        0       0
997   18           3    95825  …       0        1        0        0       0
998   18           3    95825  …       0        0        0        0       0
999   18           3    95825  …       0        0        0        0       0

     Thriller  War  Western  Year  Movie_Age
0           0    0        0  1995          5
1           0    0        0  1995          5
2           0    0        0  1995          5
3           0    0        0  1977         23
4           0    1        0  1993          7
..        …  …      …   …        …
995         0    0        0  1998          2
996         1    0        0  1998          2
997         0    0        0  1998          2
998         0    1        0  1985         15
999         0    1        0  1988         12

[1000 rows x 30 columns]
```

[36]:
```
# Use the following features:movie id,age,occupation
features = first_500[['MovieID','Age','Occupation']].values
features
```

[36]:
```
array([[   1,    1,   10],
       [  48,    1,   10],
       [ 150,    1,   10],
       …,
       [2394,   18,    3],
```

```
            [2402,   18,    3],
            [2404,   18,    3]], dtype=int64)
```

[37]:
```
# Use rating as label
labels = first_500[['Rating']].values
labels
```

[37]:
```
array([[5],
       [5],
       [5],
       [4],
       [5],
       [4],
       [4],
       [4],
       [5],
       [4],
       [3],
       [3],
       [3],
       [4],
       [3],
       [4],
       [4],
       [5],
       [5],
       [5],
       [5],
       [4],
       [5],
       [3],
       [4],
       [4],
       [5],
       [5],
       [4],
       [4],
       [4],
       [5],
       [4],
       [5],
       [4],
       [4],
       [5],
       [4],
       [3],
       [3],
```

[5],
[4],
[3],
[4],
[4],
[4],
[4],
[5],
[4],
[5],
[4],
[4],
[4],
[4],
[4],
[4],
[5],
[5],
[4],
[2],
[4],
[4],
[3],
[4],
[4],
[4],
[3],
[4],
[5],
[4],
[4],
[5],
[4],
[3],
[4],
[4],
[5],
[4],
[5],
[4],
[4],
[3],
[3],
[5],
[4],
[4],
[4],

[4],
[5],
[3],
[5],
[3],
[4],
[3],
[4],
[3],
[3],
[3],
[4],
[5],
[3],
[3],
[4],
[1],
[5],
[4],
[5],
[5],
[5],
[3],
[3],
[4],
[5],
[4],
[4],
[3],
[5],
[3],
[4],
[3],
[3],
[4],
[4],
[5],
[4],
[3],
[4],
[4],
[4],
[4],
[5],
[4],
[3],
[3],

[5],
[4],
[4],
[5],
[5],
[4],
[4],
[3],
[5],
[4],
[5],
[4],
[4],
[4],
[3],
[5],
[5],
[5],
[3],
[4],
[4],
[2],
[3],
[5],
[3],
[5],
[3],
[3],
[3],
[5],
[4],
[3],
[4],
[5],
[5],
[5],
[5],
[3],
[5],
[5],
[4],
[3],
[4],
[4],
[5],
[5],
[5],

[3],
[4],
[5],
[5],
[4],
[3],
[3],
[4],
[4],
[4],
[3],
[5],
[5],
[3],
[5],
[5],
[4],
[3],
[4],
[5],
[3],
[5],
[4],
[4],
[3],
[2],
[4],
[4],
[5],
[3],
[3],
[5],
[3],
[3],
[5],
[3],
[3],
[2],
[3],
[5],
[3],
[5],
[5],
[2],
[4],
[2],
[3],

[5],
[2],
[4],
[3],
[5],
[5],
[3],
[3],
[5],
[3],
[5],
[3],
[4],
[5],
[5],
[3],
[3],
[2],
[4],
[3],
[4],
[3],
[3],
[5],
[3],
[4],
[3],
[5],
[5],
[3],
[3],
[3],
[4],
[3],
[4],
[5],
[4],
[4],
[5],
[4],
[3],
[4],
[4],
[4],
[5],
[4],
[3],

[3],
[3],
[3],
[5],
[4],
[3],
[4],
[5],
[5],
[5],
[5],
[3],
[4],
[4],
[4],
[5],
[3],
[3],
[2],
[4],
[3],
[4],
[4],
[4],
[4],
[4],
[4],
[4],
[4],
[3],
[4],
[4],
[3],
[3],
[3],
[4],
[3],
[3],
[4],
[4],
[2],
[5],
[4],
[4],
[4],
[3],
[4],

[4],
[5],
[4],
[3],
[3],
[4],
[5],
[5],
[5],
[4],
[4],
[4],
[4],
[4],
[5],
[4],
[5],
[2],
[4],
[4],
[4],
[5],
[4],
[3],
[4],
[4],
[3],
[3],
[3],
[4],
[4],
[3],
[4],
[2],
[4],
[2],
[2],
[3],
[3],
[4],
[3],
[4],
[2],
[4],
[2],
[3],
[3],

[5],
[5],
[4],
[3],
[5],
[4],
[5],
[3],
[4],
[3],
[5],
[3],
[2],
[3],
[4],
[4],
[2],
[5],
[5],
[5],
[4],
[3],
[5],
[4],
[5],
[4],
[4],
[5],
[3],
[5],
[4],
[4],
[5],
[5],
[4],
[4],
[4],
[3],
[4],
[4],
[4],
[4],
[4],
[5],
[5],
[4],
[4],
[3],

[3],
[5],
[4],
[4],
[5],
[4],
[5],
[5],
[4],
[4],
[4],
[4],
[3],
[5],
[3],
[5],
[4],
[4],
[5],
[3],
[5],
[5],
[3],
[4],
[4],
[5],
[4],
[3],
[3],
[5],
[5],
[5],
[5],
[5],
[5],
[3],
[3],
[4],
[2],
[4],
[4],
[5],
[5],
[5],
[3],
[5],
[5],

[3],
[3],
[5],
[4],
[3],
[5],
[5],
[4],
[5],
[3],
[4],
[4],
[5],
[5],
[5],
[5],
[5],
[5],
[5],
[4],
[5],
[3],
[5],
[4],
[4],
[5],
[4],
[4],
[4],
[4],
[4],
[5],
[5],
[5],
[5],
[5],
[2],
[3],
[4],
[4],
[4],
[4],
[4],
[4],
[3],
[5],
[4],
[4],
[4],

[4],
[5],
[5],
[3],
[3],
[3],
[5],
[3],
[3],
[5],
[3],
[5],
[5],
[4],
[5],
[3],
[5],
[5],
[3],
[5],
[3],
[5],
[5],
[4],
[4],
[5],
[3],
[5],
[3],
[5],
[3],
[5],
[5],
[4],
[4],
[4],
[4],
[4],
[3],
[5],
[3],
[3],
[4],
[5],
[4],
[5],
[3],

[3],
[4],
[3],
[5],
[5],
[4],
[5],
[4],
[3],
[4],
[2],
[5],
[5],
[4],
[5],
[4],
[5],
[3],
[4],
[5],
[5],
[2],
[3],
[4],
[5],
[3],
[5],
[4],
[3],
[5],
[5],
[5],
[5],
[4],
[4],
[4],
[4],
[3],
[4],
[4],
[5],
[5],
[4],
[4],
[3],
[4],
[3],

[5],
[3],
[4],
[4],
[5],
[4],
[3],
[4],
[3],
[4],
[4],
[5],
[3],
[5],
[4],
[3],
[4],
[4],
[5],
[5],
[5],
[5],
[4],
[4],
[5],
[3],
[5],
[4],
[4],
[4],
[4],
[3],
[5],
[5],
[5],
[5],
[5],
[5],
[4],
[4],
[5],
[5],
[5],
[4],
[4],
[5],
[5],
[4],

[4],
[4],
[3],
[4],
[5],
[5],
[4],
[4],
[5],
[5],
[5],
[5],
[5],
[5],
[4],
[5],
[4],
[4],
[4],
[5],
[4],
[3],
[5],
[5],
[4],
[3],
[5],
[4],
[4],
[5],
[4],
[4],
[5],
[5],
[5],
[4],
[4],
[3],
[5],
[5],
[4],
[3],
[4],
[4],
[5],
[5],
[3],
[4],

[3],
[4],
[4],
[4],
[4],
[3],
[5],
[4],
[2],
[4],
[4],
[5],
[4],
[5],
[3],
[3],
[5],
[4],
[2],
[5],
[4],
[5],
[4],
[4],
[3],
[4],
[3],
[3],
[4],
[5],
[3],
[4],
[3],
[5],
[5],
[3],
[5],
[4],
[3],
[4],
[3],
[2],
[5],
[5],
[5],
[5],
[4],

[4],
[5],
[5],
[2],
[5],
[4],
[3],
[3],
[4],
[4],
[5],
[3],
[5],
[3],
[3],
[2],
[2],
[4],
[5],
[5],
[4],
[3],
[3],
[4],
[4],
[4],
[2],
[5],
[4],
[4],
[3],
[5],
[4],
[1],
[4],
[5],
[3],
[1],
[5],
[3],
[4],
[5],
[1],
[1],
[3],
[4],
[4],

[5],
[5],
[5],
[3],
[4],
[1],
[4],
[5],
[5],
[4],
[3],
[5],
[5],
[4],
[3],
[2],
[3],
[1],
[5],
[4],
[2],
[3],
[3],
[4],
[2],
[3],
[5],
[5],
[1],
[2],
[5],
[2],
[4],
[4],
[5],
[5],
[4],
[4],
[5],
[5],
[4],
[5],
[4],
[3],
[1],
[1],
[5],

[5],
[3],
[3],
[4],
[1],
[5],
[5],
[5],
[4],
[5],
[1],
[3],
[5],
[4],
[5],
[4],
[4],
[4],
[3],
[1],
[2],
[4],
[5],
[4],
[5],
[1],
[5],
[1],
[4],
[5],
[5],
[5],
[4],
[4],
[5],
[5],
[5],
[5],
[5],
[5],
[5],
[5],
[5],
[5],
[5],
[5],
[5],

[3],
[5],
[1],
[2],
[4],
[3],
[4],
[3],
[4],
[5],
[5],
[1],
[4],
[4],
[3],
[1],
[1],
[5],
[4],
[1],
[2],
[5],
[1],
[4],
[5],
[4],
[3],
[5],
[4],
[5],
[4],
[1],
[4],
[4],
[4],
[3],
[1],
[4],
[1],
[3],
[4],
[5],
[3],
[1],
[5],
[3],
[3],

[4],
[5],
[5],
[5],
[5],
[4],
[3],
[4],
[5],
[3],
[2],
[4],
[5],
[5],
[4],
[2],
[5],
[1],
[3],
[3],
[3],
[4],
[5],
[5],
[4],
[5],
[4],
[4],
[4],
[5],
[3],
[5],
[4],
[4],
[5],
[4],
[3],
[3],
[5],
[5],
[4],
[5],
[3],
[5],
[3],
[3],
[5],

```
       [4],
       [5],
       [4],
       [4],
       [1],
       [4],
       [3],
       [5],
       [4],
       [5],
       [3],
       [5],
       [5],
       [3],
       [4],
       [2],
       [1],
       [4],
       [2],
       [2]], dtype=int64)
```

[38]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

[39]:
```
# Create train and test data set
train, test, train_labels, test_labels =␣
 ↪train_test_split(features,labels,test_size=0.33,random_state=42)
```

[40]:
```
train
```

[40]:
```
array([[3035,   35,    1],
       [1393,   25,   17],
       [3198,   35,    1],
       …,
       [1073,   18,    3],
       [ 784,   35,    1],
       [2802,   50,    9]], dtype=int64)
```

[41]:
```
test
```

[41]:
```
array([[1265,   35,    1],
       [3408,   35,    1],
```

```
[3447,   35,    1],
[2423,   35,    1],
[ 539,   35,    1],
[2657,   35,    1],
[2109,   35,    1],
[1247,   35,    1],
[1064,   18,    3],
[ 105,   25,   12],
[ 421,   18,    3],
[1035,   50,    9],
[2140,   35,    1],
[2140,   18,    3],
[2000,   18,    3],
[1527,   18,    3],
[ 508,   25,   17],
[1275,   18,    3],
[3704,   35,    1],
[1653,   25,   17],
[1380,   35,    1],
[1027,   25,   12],
[   7,   35,    1],
[1282,   35,    1],
[2006,   25,   12],
[2712,   25,   12],
[2506,   50,    9],
[2278,   18,    3],
[1562,   18,    3],
[2021,   18,    3],
[3114,   25,   17],
[ 150,   25,   12],
[2094,   35,    1],
[1203,   35,    1],
[   2,   35,    1],
[1693,   25,   12],
[2795,   35,    1],
[1948,   35,    1],
[1552,   18,    3],
[ 296,   50,    9],
[1923,   25,   17],
[2100,   50,    9],
[1446,   25,   17],
[2141,   18,    3],
[1148,   25,   17],
[ 428,   25,   17],
[1801,   18,    3],
[2033,   35,    1],
[ 898,   35,    1],
```

```
[ 737,   18,    3],
[1676,   35,    1],
[1246,   18,    3],
[3500,   25,   12],
[1884,   35,    1],
[1197,    1,   10],
[1721,    1,   10],
[2087,   35,    1],
[ 661,    1,   10],
[2336,   25,   12],
[ 532,   18,    3],
[ 994,   25,   17],
[  17,   50,    9],
[1357,   35,    1],
[1916,   25,   12],
[2043,   35,    1],
[2863,   35,    1],
[2431,   35,    1],
[1186,   18,    3],
[ 912,   50,    9],
[1371,   35,    1],
[3408,   50,    9],
[1148,   35,    1],
[2001,   35,    1],
[3451,   35,    1],
[ 838,   25,   17],
[ 745,   25,   17],
[2375,   35,    1],
[2340,    1,   10],
[2000,   35,    1],
[1310,   25,   17],
[2662,   35,    1],
[ 590,   50,    9],
[ 377,   25,   17],
[ 595,   50,    9],
[1639,   25,   17],
[1377,   35,    1],
[2402,   18,    3],
[3108,   35,    1],
[3668,   35,    1],
[1921,   25,   17],
[ 186,   35,    1],
[ 923,   35,    1],
[1269,   35,    1],
[2320,   25,   12],
[ 168,   18,    3],
[ 802,   35,    1],
```

```
[3809,   35,    1],
[1688,   50,    9],
[ 383,   50,    9],
[ 588,   18,    3],
[3153,   35,    1],
[3253,   25,   17],
[1967,   18,    3],
[3267,   25,   12],
[2138,   35,    1],
[2302,   35,    1],
[1411,   35,    1],
[2085,   18,    3],
[ 589,   25,   12],
[1690,   18,    3],
[1283,   35,    1],
[ 552,   18,    3],
[2268,   18,    3],
[ 485,   18,    3],
[1569,   50,    9],
[ 750,   35,    1],
[1411,   25,   12],
[2153,   18,    3],
[1294,   35,    1],
[ 912,   25,   17],
[1387,   18,    3],
[ 597,   35,    1],
[3100,   35,    1],
[3481,   25,   12],
[2908,   25,   12],
[1566,   35,    1],
[1947,   35,    1],
[ 914,   35,    1],
[1193,   18,    3],
[ 302,   18,    3],
[2018,   35,    1],
[1005,   18,    3],
[  25,   25,   17],
[2384,   18,    3],
[1286,   35,    1],
[  34,   50,    9],
[3685,   50,    9],
[2291,   25,   12],
[1246,    1,   10],
[ 920,   50,    9],
[2762,    1,   10],
[3213,   25,   12],
[3194,   35,    1],
```

```
[ 524,   25,   17],
[1391,   18,    3],
[1682,   18,    3],
[ 412,   18,    3],
[3155,   25,   12],
[ 551,   18,    3],
[1682,   25,   17],
[2300,   35,    1],
[ 476,   25,   12],
[2145,   18,    3],
[ 720,   35,    1],
[2005,   18,    3],
[1009,   35,    1],
[1356,   25,   17],
[2948,   35,    1],
[ 364,   50,    9],
[1282,   18,    3],
[2009,   35,    1],
[3869,   35,    1],
[2336,   35,    1],
[2355,   35,    1],
[1185,   18,    3],
[2498,   35,    1],
[1089,   25,   17],
[2804,   35,    1],
[1500,   25,   17],
[1358,   25,   17],
[3623,   25,   17],
[1079,   35,    1],
[3524,   50,    9],
[2336,   18,    3],
[1104,   35,    1],
[3751,   25,   17],
[3260,   25,   12],
[3425,   25,   12],
[ 327,   18,    3],
[2640,   35,    1],
[1196,   35,    1],
[2529,   35,    1],
[ 110,   35,    1],
[1278,   35,    1],
[ 919,   18,    3],
[ 110,   25,   12],
[3301,   25,   17],
[3793,   25,   17],
[2006,   18,    3],
[3591,   35,    1],
```

```
[2025,   18,    3],
[ 592,   18,    3],
[2398,   35,    1],
[1701,   25,   12],
[2268,   25,   12],
[ 480,   35,    1],
[2268,   25,   17],
[1840,   25,   12],
[2078,   35,    1],
[1088,   50,    9],
[1566,    1,   10],
[1302,   35,    1],
[2918,   35,    1],
[1517,   35,    1],
[ 736,   18,    3],
[2011,   35,    1],
[ 538,   25,   12],
[1702,   18,    3],
[1197,   35,    1],
[ 555,   18,    3],
[ 588,   50,    9],
[ 161,   25,   12],
[1804,   18,    3],
[ 589,   18,    3],
[2396,   35,    1],
[2166,   25,   17],
[2453,   35,    1],
[3250,   25,   12],
[1617,   18,    3],
[3086,   35,    1],
[1265,   25,   17],
[3363,   35,    1],
[2142,   18,    3],
[3105,    1,   10],
[1411,   18,    3],
[ 150,    1,   10],
[1372,   35,    1],
[3178,   25,   17],
[1587,   18,    3],
[1304,   35,    1],
[2959,   25,   17],
[2375,   18,    3],
[1084,   35,    1],
[1756,   35,    1],
[1961,    1,   10],
[1836,    1,   10],
[2688,   25,   12],
```

```
[2093,   18,    3],
[1639,   25,   12],
[2278,   25,   17],
[ 260,    1,   10],
[1735,   25,   12],
[1250,   35,    1],
[ 296,   18,    3],
[1307,   25,   17],
[3508,   50,    9],
[ 743,   35,    1],
[1043,   50,    9],
[1441,   50,    9],
[ 597,   25,   17],
[1296,   50,    9],
[2355,   18,    3],
[ 222,   18,    3],
[1396,   18,    3],
[ 339,   35,    1],
[1721,   25,   17],
[1257,   35,    1],
[1688,   18,    3],
[1739,   18,    3],
[ 531,    1,   10],
[3347,   35,    1],
[1016,   35,    1],
[2321,   50,    9],
[  47,   25,   17],
[3452,   25,   17],
[1196,   18,    3],
[ 292,   18,    3],
[ 153,   35,    1],
[2133,   35,    1],
[ 180,   35,    1],
[1129,   35,    1],
[3624,   50,    9],
[1544,   18,    3],
[3252,   25,   12],
[1270,   35,    1],
[ 608,    1,   10],
[1678,   25,   12],
[2041,   35,    1],
[1210,   50,    9],
[ 110,   18,    3],
[1961,   35,    1],
[ 291,   18,    3],
[3006,   25,   12],
[ 778,   25,   17],
```

```
[2023,   25,   12],
[3155,   35,    1],
[3510,   25,   17],
[ 919,   35,    1],
[2541,   25,   12],
[2042,   18,    3],
[2020,   18,    3],
[ 288,   18,    3],
[1210,   35,    1],
[1015,   35,    1],
[3717,   25,   17],
[1022,   35,    1],
[1962,   18,    3],
[ 671,   35,    1],
[ 594,    1,   10],
[ 393,   25,   12],
[1356,   35,    1],
[ 899,   35,    1],
[1088,   35,    1],
[ 594,   35,    1],
[  47,   18,    3],
[2028,   18,    3],
[2294,   25,   17],
[1784,   35,    1],
[1960,   18,    3],
[1639,   35,    1],
[2006,   35,    1],
[1287,   35,    1],
[1923,   35,    1],
[3072,   35,    1],
[ 918,   35,    1],
[ 926,   35,    1],
[2571,   25,   12],
[3255,   25,   17],
[3264,   35,    1],
[2028,   25,   12],
[1101,   50,    9],
[ 282,   25,   12],
[1221,   25,   17],
[2115,   35,    1],
[  44,   18,    3],
[1223,   35,    1],
[2858,   25,   17],
[1101,   35,    1],
[3097,   35,    1],
[  42,   25,   12]], dtype=int64)
```

[42]:
```
train_labels
```

[42]:
```
array([[3],
       [3],
       [3],
       [3],
       [5],
       [5],
       [2],
       [4],
       [4],
       [5],
       [5],
       [4],
       [5],
       [3],
       [4],
       [5],
       [4],
       [5],
       [5],
       [3],
       [3],
       [3],
       [5],
       [4],
       [3],
       [4],
       [4],
       [4],
       [3],
       [5],
       [5],
       [4],
       [4],
       [5],
       [4],
       [4],
       [4],
       [5],
       [5],
       [4],
       [5],
       [4],
       [4],
       [5],
       [4],
```

[4],
[5],
[3],
[5],
[3],
[5],
[4],
[5],
[3],
[5],
[4],
[4],
[3],
[3],
[4],
[4],
[4],
[3],
[5],
[5],
[5],
[4],
[4],
[5],
[4],
[4],
[3],
[5],
[5],
[4],
[5],
[4],
[5],
[4],
[4],
[5],
[4],
[3],
[4],
[3],
[4],
[4],
[4],
[3],
[5],
[5],
[4],

[3],
[5],
[4],
[3],
[5],
[4],
[5],
[5],
[5],
[3],
[4],
[4],
[5],
[3],
[4],
[3],
[5],
[5],
[5],
[4],
[3],
[3],
[4],
[3],
[4],
[4],
[5],
[5],
[3],
[5],
[5],
[5],
[4],
[4],
[5],
[5],
[5],
[5],
[4],
[4],
[5],
[4],
[3],
[5],
[4],
[5],
[3],

[5],
[4],
[4],
[5],
[3],
[5],
[4],
[3],
[4],
[3],
[4],
[4],
[4],
[3],
[5],
[4],
[4],
[3],
[5],
[5],
[5],
[5],
[5],
[3],
[3],
[4],
[5],
[3],
[3],
[5],
[5],
[5],
[4],
[4],
[4],
[5],
[4],
[5],
[4],
[5],
[3],
[3],
[3],
[4],
[4],
[4],
[3],

[5],
[5],
[4],
[5],
[4],
[4],
[5],
[4],
[5],
[3],
[5],
[4],
[5],
[3],
[4],
[5],
[5],
[5],
[5],
[5],
[5],
[4],
[4],
[3],
[5],
[4],
[4],
[5],
[5],
[3],
[2],
[5],
[4],
[4],
[4],
[5],
[4],
[4],
[5],
[5],
[3],
[5],
[3],
[4],
[3],
[5],
[2],
[4],

[3],
[5],
[3],
[5],
[3],
[4],
[4],
[5],
[4],
[5],
[4],
[3],
[5],
[5],
[4],
[4],
[5],
[3],
[5],
[1],
[4],
[3],
[4],
[5],
[4],
[3],
[3],
[2],
[4],
[4],
[4],
[1],
[4],
[4],
[4],
[5],
[5],
[5],
[4],
[4],
[5],
[4],
[5],
[4],
[3],
[3],
[4],

[5],
[4],
[3],
[3],
[4],
[4],
[5],
[3],
[1],
[4],
[4],
[5],
[5],
[3],
[4],
[5],
[4],
[5],
[4],
[3],
[3],
[5],
[5],
[4],
[5],
[4],
[5],
[4],
[5],
[3],
[2],
[2],
[4],
[4],
[4],
[5],
[5],
[3],
[5],
[3],
[3],
[4],
[1],
[4],
[3],
[5],
[3],

[4],
[3],
[1],
[3],
[4],
[3],
[5],
[4],
[2],
[5],
[1],
[4],
[4],
[4],
[5],
[5],
[5],
[3],
[4],
[5],
[4],
[5],
[4],
[4],
[1],
[4],
[5],
[2],
[5],
[3],
[4],
[4],
[4],
[5],
[5],
[5],
[5],
[3],
[4],
[3],
[4],
[4],
[5],
[1],
[5],
[2],
[3],

[3],
[4],
[4],
[4],
[4],
[3],
[2],
[5],
[3],
[3],
[3],
[3],
[3],
[5],
[5],
[5],
[4],
[5],
[4],
[4],
[4],
[2],
[4],
[4],
[5],
[5],
[3],
[4],
[3],
[3],
[4],
[2],
[3],
[5],
[4],
[5],
[5],
[3],
[4],
[4],
[5],
[3],
[4],
[5],
[3],
[3],
[4],
[5],

[5],
[2],
[5],
[5],
[4],
[4],
[5],
[5],
[5],
[5],
[3],
[5],
[5],
[4],
[3],
[5],
[5],
[4],
[5],
[4],
[3],
[4],
[5],
[4],
[4],
[5],
[2],
[3],
[4],
[4],
[4],
[5],
[3],
[4],
[2],
[3],
[3],
[3],
[4],
[3],
[5],
[5],
[3],
[1],
[3],
[4],
[5],

[5],
[5],
[3],
[4],
[5],
[3],
[4],
[5],
[3],
[5],
[5],
[4],
[3],
[5],
[4],
[4],
[3],
[4],
[4],
[3],
[3],
[5],
[4],
[5],
[3],
[4],
[3],
[5],
[1],
[2],
[4],
[5],
[3],
[3],
[5],
[3],
[5],
[4],
[4],
[1],
[3],
[3],
[4],
[3],
[5],
[4],
[5],

[5],
[5],
[3],
[4],
[5],
[4],
[4],
[5],
[5],
[5],
[5],
[5],
[4],
[4],
[5],
[3],
[4],
[4],
[4],
[5],
[4],
[2],
[5],
[3],
[5],
[4],
[4],
[5],
[3],
[5],
[5],
[4],
[2],
[3],
[3],
[5],
[4],
[5],
[3],
[5],
[3],
[4],
[4],
[5],
[4],
[2],
[5],

[3],
[3],
[4],
[5],
[4],
[5],
[4],
[1],
[4],
[4],
[4],
[3],
[3],
[5],
[4],
[5],
[4],
[5],
[2],
[2],
[4],
[3],
[4],
[5],
[5],
[4],
[5],
[4],
[3],
[4],
[3],
[4],
[4],
[2],
[2],
[3],
[2],
[1],
[4],
[3],
[5],
[5],
[4],
[2],
[4],
[5],
[4],

[4],
[3],
[5],
[4],
[5],
[5],
[4],
[4],
[5],
[3],
[4],
[5],
[4],
[4],
[4],
[1],
[4],
[5],
[4],
[4],
[4],
[4],
[4],
[5],
[4],
[5],
[3],
[4],
[4],
[5],
[3],
[3],
[5],
[3],
[2],
[4],
[4],
[4],
[5],
[4],
[3],
[5],
[5],
[5],
[4],
[5],
[3],
[4],

```
       [5],
       [5],
       [3],
       [4],
       [4],
       [4],
       [5],
       [4],
       [5],
       [5],
       [4],
       [4],
       [3],
       [4]], dtype=int64)
```

[43]:
```
test_labels
```

[43]:
```
array([[5],
       [4],
       [5],
       [5],
       [5],
       [4],
       [4],
       [3],
       [2],
       [4],
       [4],
       [5],
       [5],
       [4],
       [4],
       [4],
       [3],
       [5],
       [2],
       [4],
       [5],
       [4],
       [4],
       [5],
       [3],
       [3],
       [3],
       [3],
       [1],
       [4],
```

[4],
[4],
[4],
[3],
[5],
[3],
[3],
[4],
[1],
[2],
[5],
[3],
[3],
[5],
[4],
[3],
[1],
[3],
[4],
[1],
[4],
[5],
[3],
[2],
[3],
[4],
[5],
[3],
[3],
[1],
[4],
[4],
[5],
[5],
[5],
[4],
[5],
[1],
[4],
[4],
[5],
[5],
[4],
[5],
[3],
[4],
[4],

[3],
[5],
[3],
[4],
[3],
[3],
[4],
[4],
[3],
[2],
[5],
[4],
[4],
[3],
[3],
[5],
[2],
[1],
[5],
[5],
[5],
[4],
[5],
[2],
[4],
[5],
[5],
[3],
[5],
[4],
[4],
[5],
[1],
[3],
[2],
[4],
[2],
[4],
[4],
[5],
[3],
[4],
[4],
[4],
[4],
[3],
[4],

[3],
[3],
[5],
[5],
[4],
[5],
[4],
[1],
[4],
[2],
[5],
[4],
[3],
[5],
[4],
[4],
[4],
[3],
[4],
[4],
[3],
[4],
[5],
[3],
[5],
[4],
[5],
[3],
[3],
[5],
[5],
[5],
[3],
[4],
[4],
[5],
[4],
[3],
[5],
[4],
[5],
[4],
[4],
[5],
[4],
[4],
[4],

[5],
[3],
[5],
[4],
[4],
[3],
[3],
[3],
[5],
[5],
[5],
[4],
[4],
[5],
[5],
[5],
[5],
[2],
[4],
[3],
[5],
[2],
[4],
[5],
[4],
[3],
[4],
[4],
[4],
[4],
[5],
[4],
[5],
[5],
[3],
[1],
[4],
[3],
[4],
[5],
[4],
[4],
[3],
[3],
[5],
[5],
[4],
[4],
[3],

[5],
[4],
[4],
[5],
[3],
[5],
[5],
[5],
[4],
[3],
[4],
[3],
[4],
[4],
[4],
[4],
[5],
[5],
[3],
[3],
[5],
[4],
[4],
[4],
[3],
[5],
[4],
[3],
[3],
[4],
[4],
[3],
[3],
[5],
[3],
[4],
[5],
[5],
[5],
[4],
[1],
[4],
[5],
[5],
[3],
[5],
[2],

[5],
[4],
[3],
[4],
[2],
[4],
[4],
[3],
[3],
[4],
[4],
[5],
[3],
[3],
[5],
[5],
[1],
[3],
[5],
[3],
[5],
[3],
[5],
[3],
[1],
[5],
[4],
[4],
[3],
[3],
[5],
[5],
[3],
[4],
[2],
[5],
[4],
[5],
[5],
[1],
[5],
[4],
[5],
[5],
[2],
[3],
[3],

```
       [5],
       [4],
       [5],
       [4],
       [5],
       [4],
       [3],
       [5],
       [4],
       [3],
       [4],
       [5],
       [4],
       [5],
       [4],
       [4],
       [5],
       [3]], dtype=int64)
```

[44]:
```
# Logistic Regression
logreg = LogisticRegression()
logreg.fit(train, train_labels)
Y_pred = logreg.predict(test)
acc_log = round(logreg.score(train, train_labels) * 100, 2)
acc_log
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:760:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

[44]:
```
36.72
```

[45]:
```
# Support Vector Machines

svc = SVC()
```

```
svc.fit(train, train_labels)
Y_pred = svc.predict(test)
acc_svc = round(svc.score(train, train_labels) * 100, 2)
acc_svc
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:760:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[45]: 38.81

[46]:
```
# K Nearest Neighbors Classifier

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(train, train_labels)
Y_pred = knn.predict(test)
acc_knn = round(knn.score(train, train_labels) * 100, 2)
acc_knn
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:4:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  after removing the cwd from sys.path.
```

[46]: 59.7

[47]:
```
# Gaussian Naive Bayes

gaussian = GaussianNB()
gaussian.fit(train, train_labels)
Y_pred = gaussian.predict(test)
acc_gaussian = round(gaussian.score(train, train_labels) * 100, 2)
acc_gaussian
```

```
/usr/local/lib/python3.7/site-packages/sklearn/naive_bayes.py:206:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[47]: 39.55

[49]:
```
# Linear SVC

linear_svc = LinearSVC()
```

```
linear_svc.fit(train, train_labels)
Y_pred = linear_svc.predict(test)
acc_linear_svc = round(linear_svc.score(train, train_labels) * 100, 2)
acc_linear_svc
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:760:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/site-packages/sklearn/svm/_base.py:947:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[49]: 27.46

[50]:
```
# Decision Tree

decision_tree = DecisionTreeClassifier()
decision_tree.fit(train, train_labels)
Y_pred = decision_tree.predict(test)
acc_decision_tree = round(decision_tree.score(train, train_labels) * 100, 2)
acc_decision_tree
```

[50]: 100.0

[51]:
```
# Random Forest

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(train, train_labels)
Y_pred = random_forest.predict(test)
random_forest.score(train, train_labels)
acc_random_forest = round(random_forest.score(train, train_labels) * 100, 2)
acc_random_forest
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:4:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  after removing the cwd from sys.path.
```

[51]: 100.0

[52]:
```
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Linear SVC', 'Decision Tree'],
```

```
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest,acc_gaussian, acc_linear_svc,␣
 ↪acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

[52]:

	Model	Score
3	Random Forest	100.00
6	Decision Tree	100.00
1	KNN	59.70
4	Naive Bayes	39.55
0	Support Vector Machines	38.81
2	Logistic Regression	36.72
5	Linear SVC	27.46