Movie Lens

May 24, 2020

MOVIE LENS PROJECT ANALYSIS

1. Prepare Problem

[1]:
```
# a) Load libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

[2]:
```
# b) Load dataset
movie_data = pd.read_csv("/Users/ishanmishra/Downloads/DS with Python/final␣
↪project/Data-Science-with-Python-Project-One-master/Data science with Python␣
↪1/movies.dat",
                         sep="::", header=None,␣
↪names=['MovieID','Title','Genres'],
                         dtype={'MovieID': np.int32, 'Title': np.str, 'Genres':␣
↪np.str}, engine='python')
users_data = pd.read_csv("/Users/ishanmishra/Downloads/DS with Python/final␣
↪project/Data-Science-with-Python-Project-One-master/Data science with Python␣
↪1/users.dat",
                         sep="::", header=None,␣
↪names=['UserID','Gender','Age','Occupation','Zip-code'],
```

```
                      dtype={'UserID': np.int32, 'Gender': np.str, 'Age': np.
↪int32, 'Occupation' : np.int32, 'Zip-code' : np.str}, engine='python')
ratings_data = pd.read_csv("/Users/ishanmishra/Downloads/DS with Python/final␣
↪project/Data-Science-with-Python-Project-One-master/Data science with Python␣
↪1/ratings.dat",
                      sep="::", header=None,␣
↪names=['UserID','MovieID','Rating','Timestamp'],
                dtype={'UserID': np.int32, 'MovieID': np.int32, 'Rating': np.
↪int32, 'Timestamp' : np.str}, engine='python')
```

0.1 2. Summarize Data

[3]:
```
movie_data.head()
```

[3]:

	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

[4]:
```
movie_data.shape
```

[4]:
```
(3883, 3)
```

[5]:
```
movie_data.isnull().sum()
```

[5]:
```
MovieID    0
Title      0
Genres     0
dtype: int64
```

[6]:
```
movie_data.describe()
```

[6]:

	MovieID
count	3883.000000
mean	1986.049446
std	1146.778349
min	1.000000
25%	982.500000
50%	2010.000000
75%	2980.500000
max	3952.000000

[7]:
```
movie_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
MovieID    3883 non-null int32
Title      3883 non-null object
Genres     3883 non-null object
dtypes: int32(1), object(2)
memory usage: 75.9+ KB
```

[8]:
```
# On users data
users_data.shape
```

[8]:
```
(6040, 5)
```

[9]:
```
users_data.head()
```

[9]:

	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

[10]:
```
users_data.describe()
```

[10]:

	UserID	Age	Occupation
count	6040.000000	6040.000000	6040.000000
mean	3020.500000	30.639238	8.146854
std	1743.742145	12.895962	6.329511
min	1.000000	1.000000	0.000000
25%	1510.750000	25.000000	3.000000
50%	3020.500000	25.000000	7.000000
75%	4530.250000	35.000000	14.000000
max	6040.000000	56.000000	20.000000

[11]:
```
users_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6040 entries, 0 to 6039
Data columns (total 5 columns):
UserID        6040 non-null int32
Gender        6040 non-null object
Age           6040 non-null int32
Occupation    6040 non-null int32
Zip-code      6040 non-null object
dtypes: int32(3), object(2)
memory usage: 165.2+ KB
```

[12]:
```
users_data.isnull().sum()
```

[12]:
```
UserID        0
Gender        0
Age           0
Occupation    0
Zip-code      0
dtype: int64
```

[13]:
```
# On Ratings data
ratings_data.head()
```

[13]:

	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

[14]:
```
ratings_data.shape
```

[14]:
```
(1000209, 4)
```

[15]:
```
ratings_data.describe()
```

[15]:

	UserID	MovieID	Rating
count	1.000209e+06	1.000209e+06	1.000209e+06
mean	3.024512e+03	1.865540e+03	3.581564e+00
std	1.728413e+03	1.096041e+03	1.117102e+00
min	1.000000e+00	1.000000e+00	1.000000e+00
25%	1.506000e+03	1.030000e+03	3.000000e+00
50%	3.070000e+03	1.835000e+03	4.000000e+00
75%	4.476000e+03	2.770000e+03	4.000000e+00
max	6.040000e+03	3.952000e+03	5.000000e+00

[16]:
```
ratings_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000209 entries, 0 to 1000208
Data columns (total 4 columns):
UserID       1000209 non-null int32
MovieID      1000209 non-null int32
Rating       1000209 non-null int32
Timestamp    1000209 non-null object
dtypes: int32(3), object(1)
memory usage: 19.1+ MB
```

[17]:
```
ratings_data.isnull().sum()
```

[17]:
```
UserID       0
MovieID      0
Rating       0
Timestamp    0
dtype: int64
```

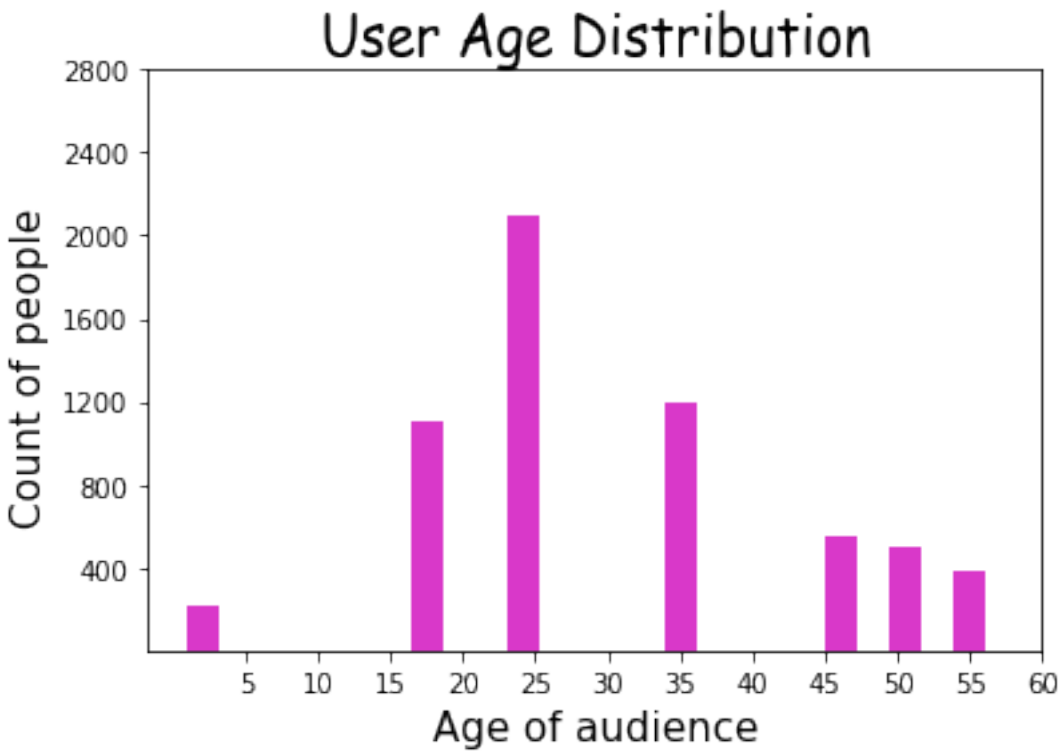
0.2 3. Data Visualizations

1 User Age Distribution

[18]:
```
age_group = users_data.groupby('Age').size()
age_group
```

[18]:
```
Age
1      222
18    1103
25    2096
35    1193
45     550
50     496
56     380
dtype: int64
```

[19]:
```
plt.hist(data=age_group,x=[users_data.Age], bins=25, color='#d938c9')
plt.title('User Age Distribution', fontdict={'fontname': 'Comic Sans MS',␣
↪'fontsize': 20})
plt.xlabel('Age of audience', fontdict={'fontsize':15})
plt.ylabel('Count of people ', fontdict={'fontsize': 15})
plt.xticks([5,10,15,20,25,30,35,40,45,50,55,60])
plt.yticks([400,800,1200,1600,2000,2400,2800])

plt.show()
```



The above age distribution shows that most of the users are 25 years old

2 Gender Distribution

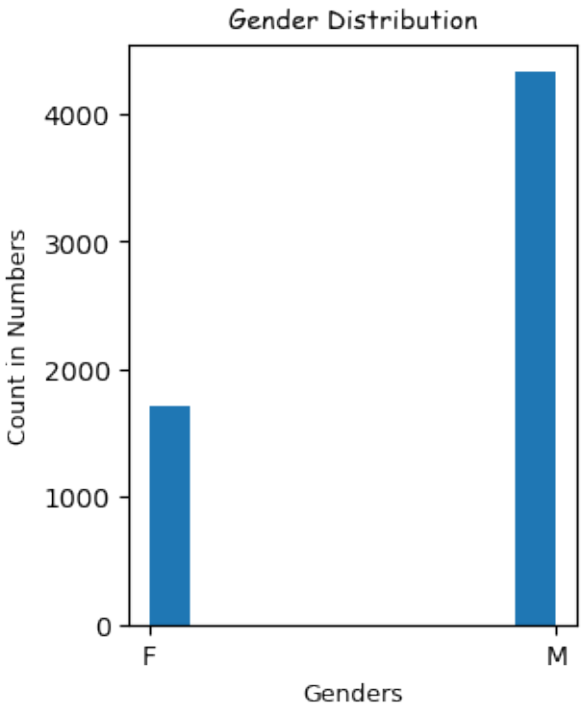
[20]:
```
gender_group = users_data.groupby('Gender').size()
gender_group
```

[20]:
```
Gender
F    1709
M    4331
dtype: int64
```

The above distribution shows that most of the users are Males

[21]:
```
plt.figure(figsize=(3,4), dpi= 100)
plt.hist(data=gender_group,x=[users_data.Gender])
plt.title('Gender Distribution', fontdict={'fontname': 'Comic Sans MS',␣
↪'fontsize': 10})
plt.xlabel('Genders', fontdict={'fontsize':9})
plt.ylabel('Count in Numbers', fontdict={'fontsize':9})
```

```
plt.show()
```

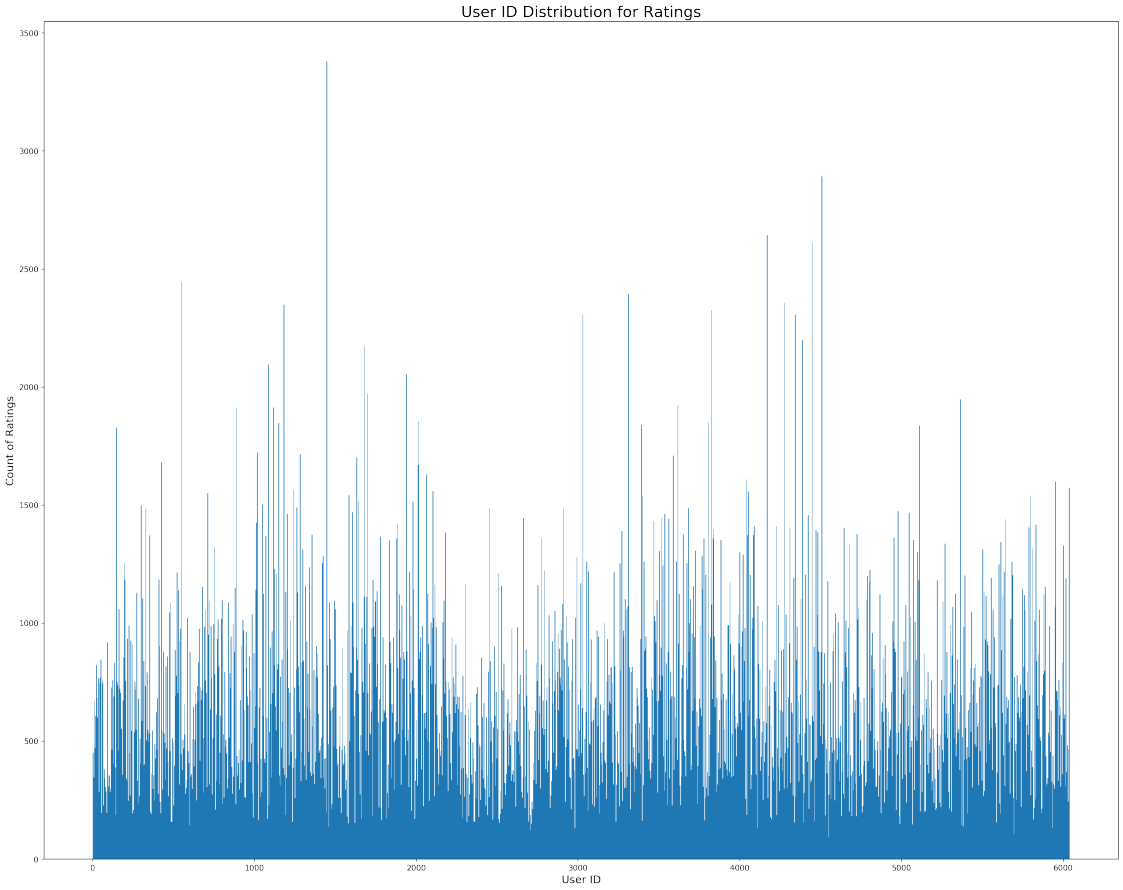


3 User Ratings

[22]:
```
user_group = ratings_data.groupby(['UserID']).size()
user_group.head(10)
```

[22]:
```
UserID
1      53
2     129
3      51
4      21
5     198
6      71
7      31
8     139
9     106
10    401
dtype: int64
```

[61]:
```
plt.figure(figsize=(25,20), dpi= 150)
plt.hist(x=[ratings_data.UserID], bins=1500)
plt.title('User ID Distribution for Ratings', fontdict={'fontsize': 20})
plt.xlabel('User ID', fontdict={'fontsize':14})
plt.ylabel('Count of Ratings', fontdict={'fontsize':14})
plt.show()
```



3.1 Toystory data

[24]:
```
toystory_data = ratings_data[ratings_data.MovieID==1]
toystory_data.head(10)
```

[24]:

	UserID	MovieID	Rating	Timestamp
40	1	1	5	978824268
469	6	1	4	978237008
581	8	1	4	978233496
711	9	1	5	978225952
837	10	1	5	978226474

```
1966       18          1          4   978154768
2276       19          1          5   978555994
2530       21          1          3   978139347
2870       23          1          4   978463614
3405       26          1          3   978130703
```

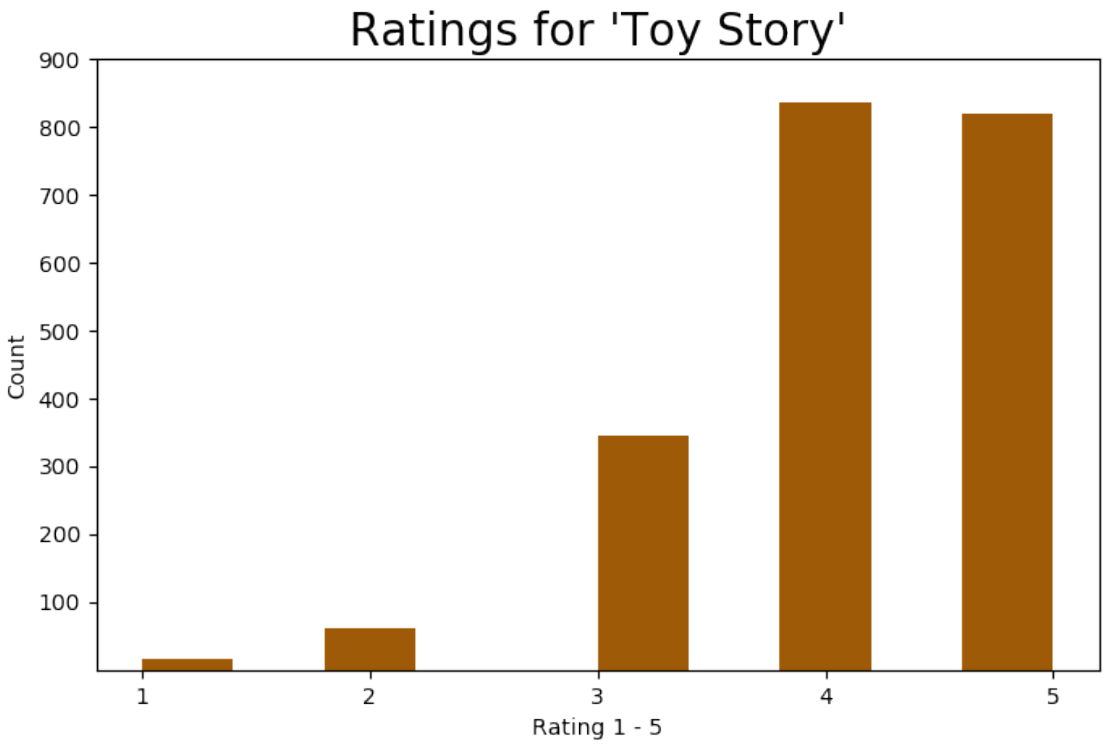
[25]:
```
toystory_data.groupby('Rating').size()
```

[25]:
```
Rating
1     16
2     61
3    345
4    835
5    820
dtype: int64
```

[26]:
```
toystory_data_group = toystory_data.groupby('Rating')
toystory_data_group
toystory_data_group.agg({'Rating':'mean'})
```

[26]:
```
        Rating
Rating
1            1
2            2
3            3
4            4
5            5
```

[27]:
```
plt.figure(figsize=(8,5), dpi= 100)
plt.hist(x=toystory_data['Rating'], color= '#9e5a06')
plt.title("Ratings for 'Toy Story'", fontdict={'fontsize':20})
plt.xlabel('Rating 1 - 5')
plt.ylabel('Count')
plt.xticks([1,2,3,4,5])
plt.yticks([100,200,300,400,500,600,700,800,900])
plt.show()
```



3.1.1 The above plot shows that the movie 'Toystory' has got 4 ** (stars) maximum #### The average rating of this movie is ## Viewership by Age for Toystory

[28]:
```
viewership = pd.merge(ratings_data, users_data, how='left', left_on=['UserID'],␣
↪right_on=['UserID'])
viewership.head()
```

[28]:

	UserID	MovieID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	1193	5	978300760	F	1	10	48067
1	1	661	3	978302109	F	1	10	48067
2	1	914	3	978301968	F	1	10	48067
3	1	3408	4	978300275	F	1	10	48067
4	1	2355	5	978824291	F	1	10	48067

[29]:
```
viewership.shape
```

[29]: (1000209, 8)

[30]:
```
ratings_data.shape
```

[30]: (1000209, 4)

[31]:
```
#select only 'Toystory' data
viewership_of_toystory = viewership[viewership['MovieID'] == 1]
viewership_of_toystory.shape
```

[31]:
```
(2077, 8)
```

[32]:
```
viewership_of_toystory.head()
```

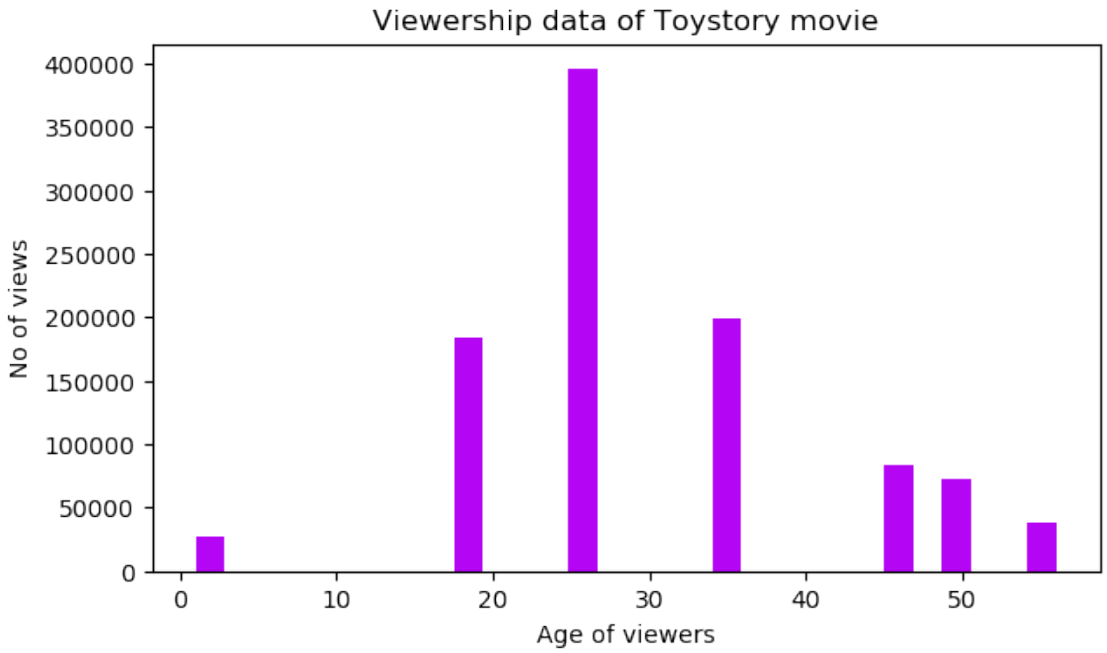
[32]:

	UserID	MovieID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
40	1	1	5	978824268	F	1	10	48067
469	6	1	4	978237008	F	50	9	55117
581	8	1	4	978233496	M	25	12	11413
711	9	1	5	978225952	M	25	17	61614
837	10	1	5	978226474	F	35	1	95370

[33]:
```
Age_of_viewers=viewership_of_toystory.groupby('Age').size()
Age_of_viewers
```

[33]:
```
Age
1     112
18    448
25    790
35    423
45    143
50    108
56     53
dtype: int64
```

[34]:
```
plt.figure(figsize=(7,4), dpi= 100)
plt.hist(data= Age_of_viewers, x= [viewership.Age], bins=30, color='#b505f5')
plt.xlabel("Age of viewers")
plt.ylabel("No of views")
plt.title("Viewership data of Toystory movie")
plt.show()
```



3.1.2 The above plot shows that the Toystory movie is more popular for viewers between Age group 20-25 years

3.2 Top 25 movies by viewership rating

[35]:
```
movie_rating = ratings_data.groupby(['MovieID'], as_index=False)
average_movie_ratings = movie_rating.agg({'Rating':'mean'})
top_25_movies = average_movie_ratings.sort_values('Rating', ascending=False).
 ↪head(25)
top_25_movies
```

[35]:

	MovieID	Rating
926	989	5.000000
3635	3881	5.000000
1652	1830	5.000000
3152	3382	5.000000
744	787	5.000000
3054	3280	5.000000
3367	3607	5.000000
3010	3233	5.000000
2955	3172	5.000000
3414	3656	5.000000
3021	3245	4.800000
51	53	4.750000

```
2309     2503  4.666667
2698     2905  4.608696
1839     2019  4.560510
309       318  4.554558
802       858  4.524966
708       745  4.520548
49         50  4.517106
513       527  4.510417
1066     1148  4.507937
2117     2309  4.500000
1626     1795  4.500000
2287     2480  4.500000
425       439  4.500000
```

[36]:
```
#The below list shows top 25 movies by viewership data
pd.merge(top_25_movies, movie_data, how='left', left_on=['MovieID'],␣
 ↪right_on=['MovieID'])
```

[36]:
```
    MovieID    Rating                                                Title  \
0       989  5.000000          Schlafes Bruder (Brother of Sleep) (1995)
1      3881  5.000000                              Bittersweet Motel (2000)
2      1830  5.000000                               Follow the Bitch (1998)
3      3382  5.000000                                Song of Freedom (1936)
4       787  5.000000                 Gate of Heavenly Peace, The (1995)
5      3280  5.000000                                     Baby, The (1973)
6      3607  5.000000                              One Little Indian (1973)
7      3233  5.000000                                  Smashing Time (1967)
8      3172  5.000000                               Ulysses (Ulisse) (1954)
9      3656  5.000000                                         Lured (1947)
10     3245  4.800000                I Am Cuba (Soy Cuba/Ya Kuba) (1964)
11       53  4.750000                                     Lamerica (1994)
12     2503  4.666667                               Apple, The (Sib) (1998)
13     2905  4.608696                                    Sanjuro (1962)
14     2019  4.560510  Seven Samurai (The Magnificent Seven) (Shichin…
15      318  4.554558                    Shawshank Redemption, The (1994)
16      858  4.524966                               Godfather, The (1972)
17      745  4.520548                               Close Shave, A (1995)
18       50  4.517106                          Usual Suspects, The (1995)
19      527  4.510417                             Schindler's List (1993)
20     1148  4.507937                          Wrong Trousers, The (1993)
21     2309  4.500000         Inheritors, The (Die Siebtelbauern) (1998)
22     1795  4.500000                Callej n de los milagros, El (1995)
23     2480  4.500000             Dry Cleaning (Nettoyage  sec) (1997)
24      439  4.500000                               Dangerous Game (1993)

                      Genres
0                      Drama
```

```
1                  Documentary
2                       Comedy
3                        Drama
4                  Documentary
5                       Horror
6         Comedy|Drama|Western
7                       Comedy
8                    Adventure
9                        Crime
10                       Drama
11                       Drama
12                       Drama
13            Action|Adventure
14                Action|Drama
15                       Drama
16          Action|Crime|Drama
17  Animation|Comedy|Thriller
18              Crime|Thriller
19                   Drama|War
20            Animation|Comedy
21                       Drama
22                       Drama
23                       Drama
24                       Drama
```

4 Rating of userid = 2696

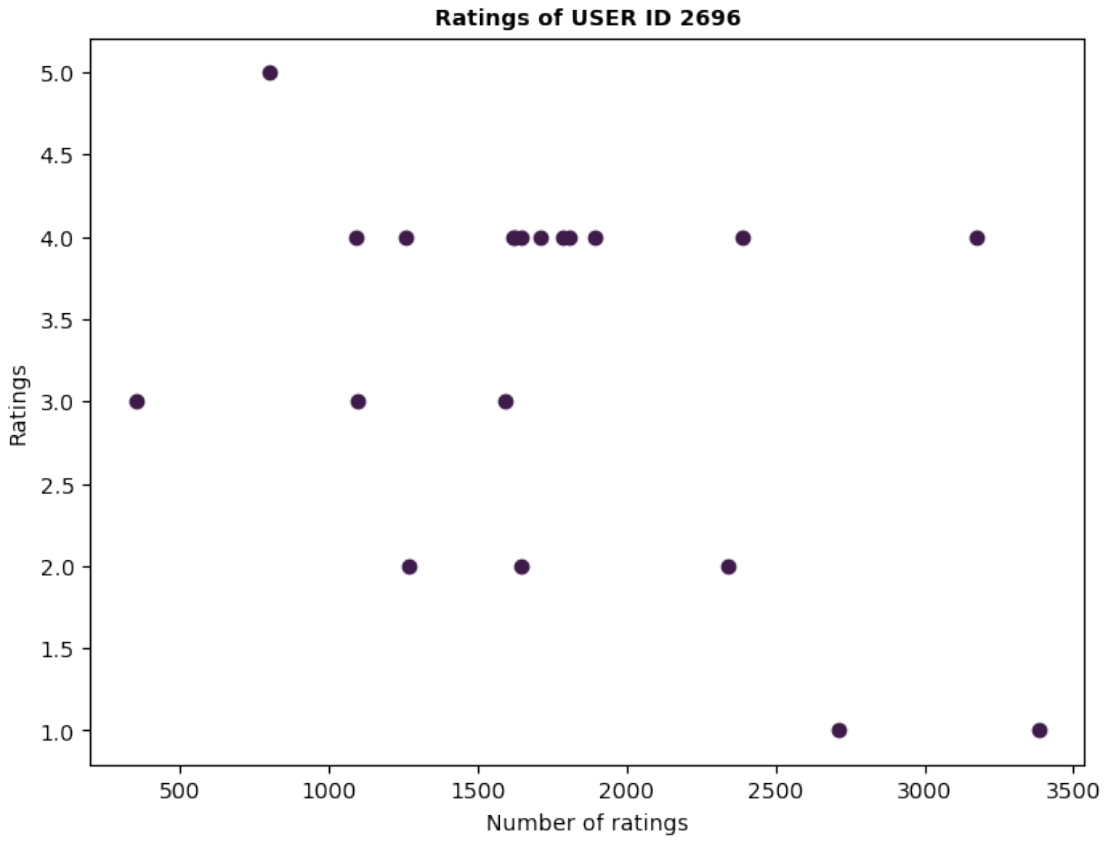
[37]:
```
users_rating_data = ratings_data[ratings_data['UserID']==2696]
users_rating_data.head(100)
```

[37]:

	UserID	MovieID	Rating	Timestamp
440667	2696	1258	4	973308710
440668	2696	1270	2	973308676
440669	2696	1617	4	973308842
440670	2696	1625	4	973308842
440671	2696	1644	2	973308920
440672	2696	1645	4	973308904
440673	2696	1805	4	973308886
440674	2696	1892	4	973308904
440675	2696	800	5	973308842
440676	2696	2338	2	973308920
440677	2696	1711	4	973308904
440678	2696	3176	4	973308865
440679	2696	2389	4	973308710
440680	2696	1589	3	973308865

```
440681    2696    2713       1  973308710
440682    2696    3386       1  973308842
440683    2696    1783       4  973308865
440684    2696     350       3  973308886
440685    2696    1092       4  973308886
440686    2696    1097       3  973308690
```

[64]:
```
# plotting the above data
plt.figure(figsize=(8,6), dpi= 100)
plt.scatter(x=users_rating_data['MovieID'], y=users_rating_data['Rating'],␣
 ↪color='#3e1b4a')
plt.xlabel("Number of ratings")
plt.ylabel("Ratings")
plt.title("Ratings of USER ID 2696", fontdict={'fontweight': 'bold', 'fontsize':
 ↪ 10})
plt.show()
```

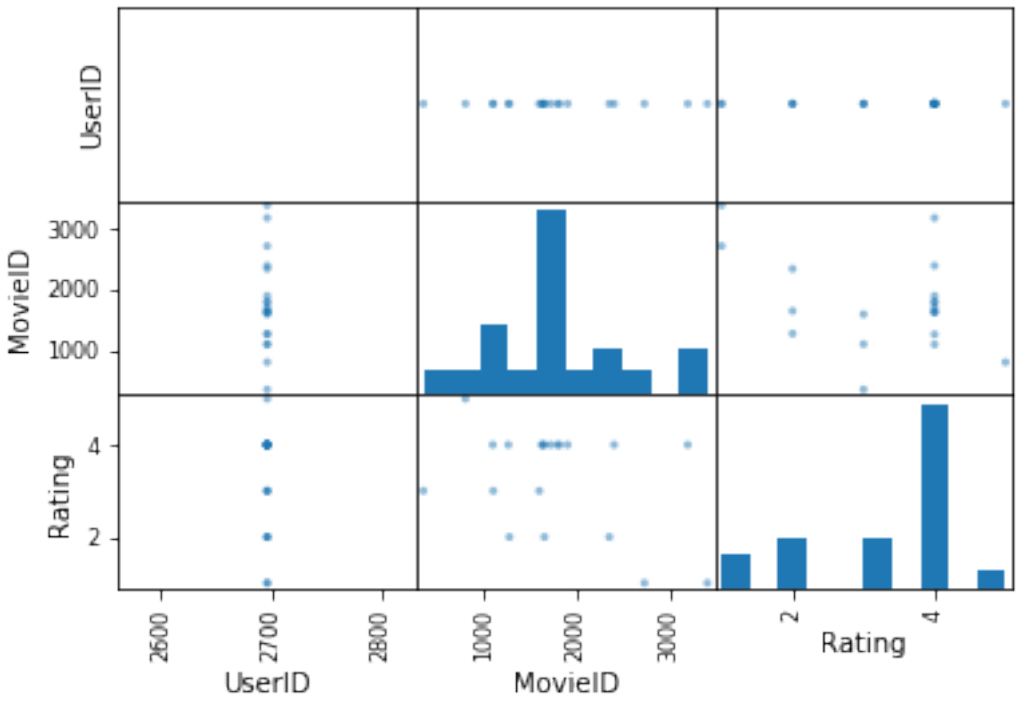


[39]:
```
from pandas.plotting import scatter_matrix
plt.figure(figsize=(7,5))
scatter_matrix(users_rating_data)
```

```
plt.show()
```

```
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/pandas/plotting/_misc.py:100: UserWarning: Attempting to set identical
left == right == 2696.0 results in singular transformations; automatically
expanding.
  ax.set_xlim(boundaries_list[i])
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/pandas/plotting/_misc.py:109: UserWarning: Attempting to set identical
bottom == top == 2696.0 results in singular transformations; automatically
expanding.
  ax.set_ylim(boundaries_list[i])
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/pandas/plotting/_misc.py:108: UserWarning: Attempting to set identical
left == right == 2696.0 results in singular transformations; automatically
expanding.
  ax.set_xlim(boundaries_list[j])
```

```
<Figure size 504x360 with 0 Axes>
```



5 Prepare Data

[40]:
```
few_viewership = viewership.head(2500)
few_viewership.shape
```

[40]:
```
(2500, 8)
```

[41]:
```
few_viewership.head()
```

[41]:

	UserID	MovieID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	1193	5	978300760	F	1	10	48067
1	1	661	3	978302109	F	1	10	48067
2	1	914	3	978301968	F	1	10	48067
3	1	3408	4	978300275	F	1	10	48067
4	1	2355	5	978824291	F	1	10	48067

[42]:
```
# preprocess data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

le.fit(few_viewership['Age'])
x_age = le.transform(few_viewership['Age'])
x_age
```

[42]:
```
array([0, 0, 0, …, 0, 0, 0])
```

[43]:
```
le.fit(few_viewership['Occupation'])
x_occ = le.transform(few_viewership['Occupation'])
x_occ
```

[43]:
```
array([5, 5, 5, …, 5, 5, 5])
```

[44]:
```
le.fit(few_viewership['MovieID'])
x_movieid = le.transform(few_viewership['MovieID'])
x_movieid
```

[44]:
```
array([ 337,  200,  247, …,  664,  361, 1159])
```

[45]:
```
few_viewership['New Age'] = x_age
few_viewership['New Occupation'] = x_occ
few_viewership['New MovieID'] = x_movieid
```

```
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy

/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[46]:
```
# Feature Selection
x_input = few_viewership[['New Age','New Occupation','New MovieID']]
y_target = few_viewership['Rating']
```

[47]:
```
x_input.head()
```

[47]:

	New Age	New Occupation	New MovieID
0	0	5	337
1	0	5	200
2	0	5	247
3	0	5	1045
4	0	5	734

6 Evaluate Algorithms

[48]:
```
# Split-out validation dataset
x_train, x_test, y_train, y_test = train_test_split(x_input, y_target,␣
 ↪test_size=0.3)
```

[49]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

[49]:
```
((1750, 3), (750, 3), (1750,), (750,))
```

[50]:
```
from sklearn.linear_model import LogisticRegression
logitReg = LogisticRegression()
lm = logitReg.fit(x_train, y_train)
```

```
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
```

[51]:
```
result = logitReg.predict(x_test)
result
```

[51]:
```
array([4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4,
       4, 5, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
       4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5,
       4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4,
```

```
       4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4,
       4, 4], dtype=int32)
```

[52]:
```
estimated = pd.Series(result, name='Estimated Values')
estimated
```

[52]:
```
0      4
1      4
2      4
3      4
4      4
5      4
6      4
7      4
8      4
9      4
10     4
11     4
12     4
13     4
14     4
15     4
16     4
17     4
18     4
19     4
20     4
21     4
22     4
23     4
24     4
25     4
26     4
27     4
28     4
29     4
      ..
720    4
721    4
722    4
723    4
724    4
725    4
726    4
```

```
727    4
728    4
729    4
730    5
731    4
732    4
733    4
734    4
735    4
736    4
737    4
738    4
739    4
740    5
741    4
742    4
743    4
744    4
745    4
746    4
747    4
748    4
749    4
Name: Estimated Values, Length: 750, dtype: int32
```

[53]:
```
final_result = pd.concat([y_test, estimated], axis=1)
```

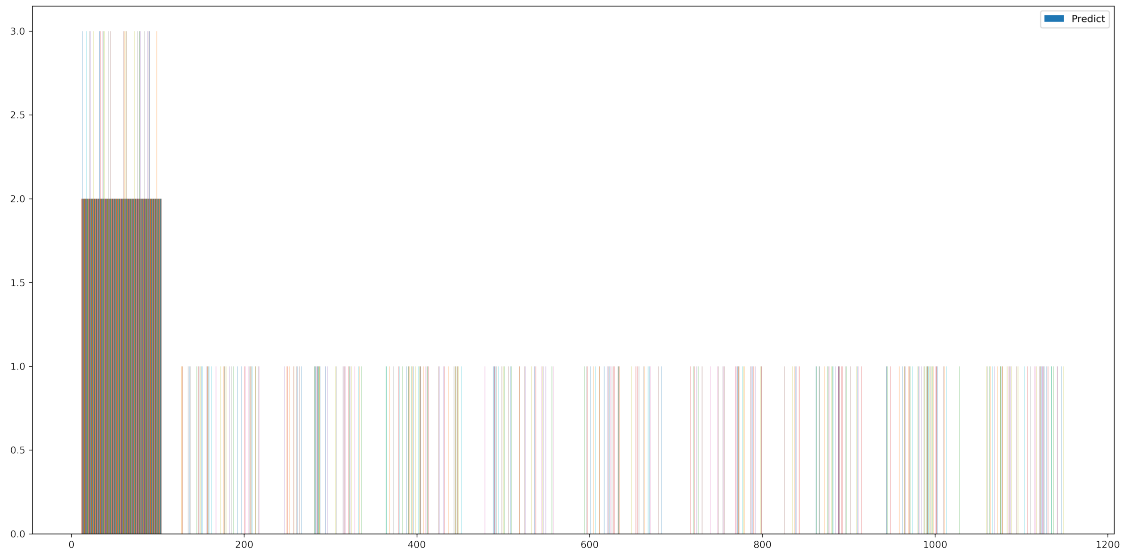
[54]:
```
# Test options and evaluation metric
print (accuracy_score(y_test, result))
print (confusion_matrix(y_test, result))
print (classification_report(y_test, result))
```

```
0.3413333333333333
[[  0   0   0  28   4]
 [  0   0   0  67   0]
 [  0   0   0 199   8]
 [  0   0   0 246  10]
 [  0   0   0 178  10]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        32
           2       0.00      0.00      0.00        67
           3       0.00      0.00      0.00       207
           4       0.34      0.96      0.51       256
           5       0.31      0.05      0.09       188

    accuracy                           0.34       750
```

```
   macro avg       0.13      0.20      0.12       750
weighted avg       0.20      0.34      0.20       750

/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
```

7 Accuracy of the above matrix is 37.2 %

[55]:
```
# Plot the histogram
plt.figure(figsize=(20,10), dpi= 300)
plt.hist(x=x_input, label= 'Predict')
plt.legend()
plt.show()
```



[56]:
```
# Spot-Check Algorithms
seed = 100
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
```

```
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed)
    cv_results = cross_val_score(model, x_train, y_train, cv=kfold,␣
 ↪scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
```

```
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
```

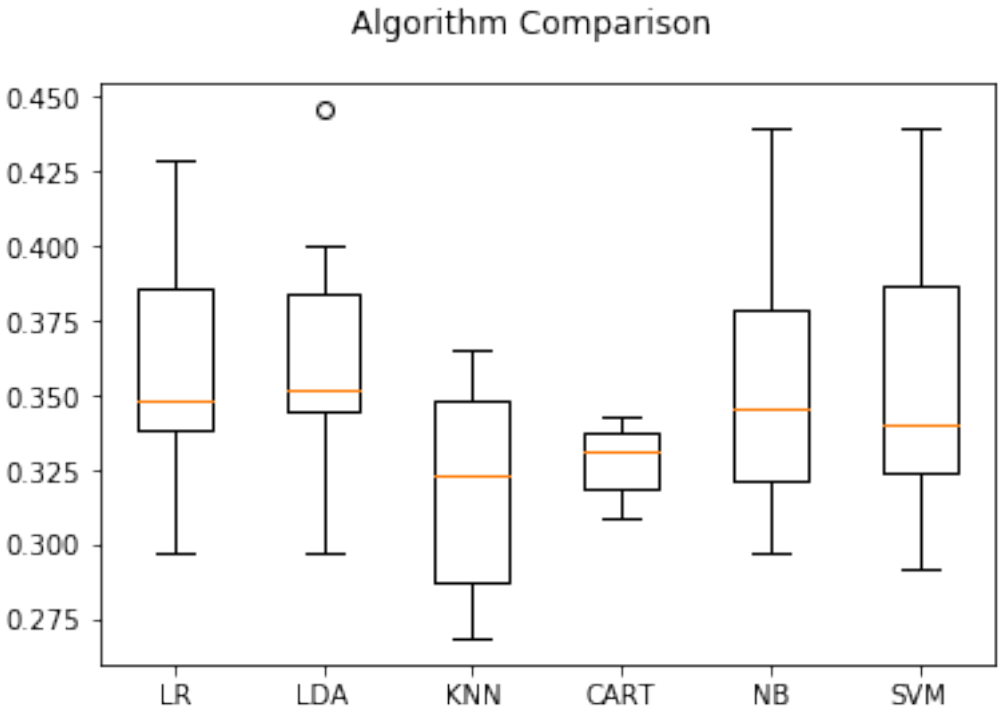
```
  FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
```

```
LR: 0.360571 (0.040037)
LDA: 0.361143 (0.039820)
KNN: 0.318286 (0.033520)
CART: 0.328000 (0.012041)
NB: 0.353143 (0.042747)
```

```
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
```

```
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/ishanmishra/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)

SVM: 0.352571 (0.046707)
```

[57]:
```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



[ ]: