Project 04: Movielens Dataset Analysis

You don't need to limit yourself to the number of rows/cells provided. You can add additional rows in each section to add more lines of code.

Happy coding!

Importing the libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
```

Importing the movies.dat dataset

In [2]:

```
dbmovies = pd.read_table('movies.dat',sep = '::',header = None)
```

In [3]:

```
# Assigning the columns names to the movies dataset
```

In [4]:

```
m_column = ['Id','Title','Genre']
```

In [5]:

```
m_column
```

Out[5]:

```
['Id', 'Title', 'Genre']
```

In [6]:

```
dbmovies.columns = m_column
```

Movies dataset after assigning the columns

In [7]:

```
dbmovies.head()
```

Out[7]:

	Id	Title	Genre
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

In [8]:

```
# unique count in the movies dataset
dbmovies.nunique()
```

Out[8]:

```
Id       3883
Title    3883
Genre     301
dtype: int64
```

Importing the rating dataset

In [9]:

```
dbrating = pd.read_table("ratings.dat",sep = '::',header = None)
```

In [10]:

```
r_col = ['Id','Review','Rating',"MobileNo"]
```

In [11]:

```
dbrating.columns = r_col
```

Rating dataset

In [12]:

```
dbrating.head()
```

Out[12]:

	Id	Review	Rating	MobileNo
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

In [13]:

```
# Number of unique element in the dataset columnwise
dbrating.nunique()
```

Out[13]:

```
Id            6040
Review        3706
Rating           5
MobileNo    458455
dtype: int64
```

Importing the users.dat dataset

In [14]:

```
dbusers = pd.read_table("users.dat",sep = '::',header = None)
```

In [15]:

```
u_col = ['Id','Gender','Age','X','occupation']
```

In [16]:

```
dbusers.columns = u_col
```

Users dataset after assigning the column_name

In [17]:

```
dbusers.head()
```

Out[17]:

	Id	Gender	Age	X	occupation
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

In [18]:

```
dbusers.nunique()
```

Out[18]:

```
Id            6040
Gender           2
Age              7
X               21
occupation    3439
dtype: int64
```

Merging of two dataset movies and users

In [19]:

```
t_mer = pd.merge(dbmovies,dbusers, how = 'inner',on = 'Id')
```

In [20]:

```
t_mer.head()
```

Out[20]:

	Id	Title	Genre	Gender	Age	X	occupation
0	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067
1	2	Jumanji (1995)	Adventure|Children's|Fantasy	M	56	16	70072
2	3	Grumpier Old Men (1995)	Comedy|Romance	M	25	15	55117
3	4	Waiting to Exhale (1995)	Comedy|Drama	M	45	7	02460
4	5	Father of the Bride Part II (1995)	Comedy	M	25	20	55455

In [21]:

```
# Checking for null values
t_mer.isnull().sum()
```

Out[21]:

```
Id            0
Title         0
Genre         0
Gender        0
Age           0
X             0
occupation    0
dtype: int64
```

The master dataset of all dataset i.e movies,users,rating

In [22]:

```
master = pd.merge(t_mer,dbrating, how = 'inner',on = "Id")
```

Final master dataset

In [23]:

```
master.head()
```

Out[23]:

	Id	Title	Genre	Gender	Age	X	occupation	Review	Rating	M
0	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	1193	5	97
1	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	661	3	97
2	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	914	3	97
3	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	3408	4	97
4	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	2355	5	97

In [24]:

```
# shape of the master dataset
master.shape
```

Out[24]:

```
(645833, 10)
```

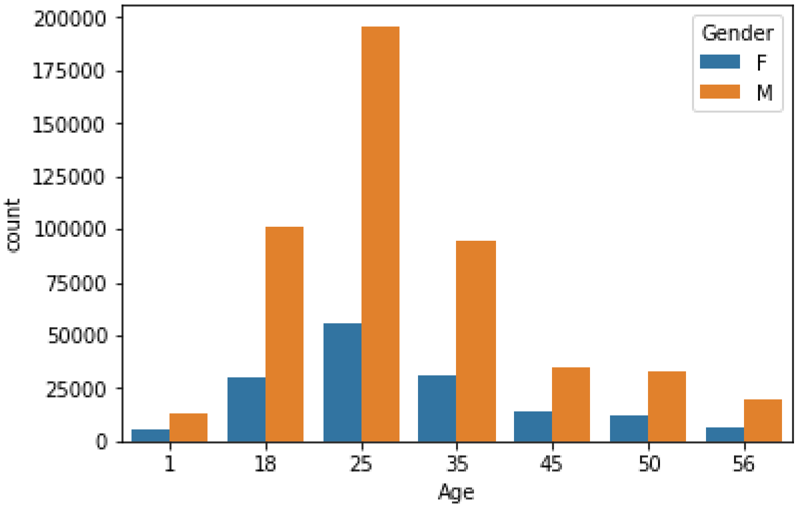
Data Understaing and Exploration

User Age Distribution

>>>The countplot show the age distribution w.r.t to Gender which show that age = 25 male is interested in watching movies and gives review with respective to the movies.

In [25]:

```
ax = sns.countplot('Age',data = master,hue = 'Gender')
```



User rating of the movie “Toy Story”

The User gives Rating to the Movies Toy Story in the year 1995

In [26]:

```
User_rating=master[master.Title == 'Toy Story (1995)'][['Title','Rating']]
```

In [27]:

```
User_rating.head()
```

Out[27]:

	Title	Rating
0	Toy Story (1995)	5
1	Toy Story (1995)	3
2	Toy Story (1995)	3
3	Toy Story (1995)	4
4	Toy Story (1995)	5

Top 25 movies by viewership rating

The top 25 most review by the user movies are following

In [28]:

```
x =master.groupby(['Title']).sum()
```

In [29]:

```
y = x.sort_values(by = ['Review'],ascending = False).head(25)
```

In [30]:

```
Viewship = pd.DataFrame(y.Review)
```

In [31]:

```
Viewship.head(25)
```

Out[31]:

	Review
Title	
Sliding Doors (1998)	3484153
Hamlet (1948)	3220078
1-900 (1994)	3014306
Shooter, The (1995)	2950602
Small Time Crooks (2000)	2549095
Seventh Heaven (Le Septi�me ciel) (1997)	2522136
Blue Chips (1994)	2479950
Waiting for Guffman (1996)	2477592
Five Wives, Three Secretaries and Me (1998)	2444231
Homeward Bound: The Incredible Journey (1993)	2413720
Air America (1990)	2409390
Friday the 13th Part VII: The New Blood (1988)	2380633
Return of Martin Guerre, The (Retour de Martin Guerre, Le) (1982)	2357811
Heathers (1989)	2312862
Thirty-Two Short Films About Glenn Gould (1993)	2277285
Dirty Dancing (1987)	2193241
Two Women (La Ciociara) (1961)	2185810
Who's That Girl? (1987)	2119146
Autumn in New York (2000)	2118298
Woman in the Dunes (Suna no onna) (1964)	2059618
Some Folks Call It a Sling Blade (1993)	2013461
Fire on the Mountain (1996)	1965215
Vermont Is For Lovers (1992)	1956705
Lawn Dogs (1997)	1945308
Big Combo, The (1955)	1922223

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

In [32]:

```
master[master.Id == 2696][['Title','Rating']]
```

Out[32]:

	Title	Rating
430638	Dinner Game, The (Le D�ner de cons) (1998)	4
430639	Dinner Game, The (Le D�ner de cons) (1998)	2
430640	Dinner Game, The (Le D�ner de cons) (1998)	4
430641	Dinner Game, The (Le D�ner de cons) (1998)	4
430642	Dinner Game, The (Le D�ner de cons) (1998)	2
430643	Dinner Game, The (Le D�ner de cons) (1998)	4
430644	Dinner Game, The (Le D�ner de cons) (1998)	4
430645	Dinner Game, The (Le D�ner de cons) (1998)	4
430646	Dinner Game, The (Le D�ner de cons) (1998)	5
430647	Dinner Game, The (Le D�ner de cons) (1998)	2
430648	Dinner Game, The (Le D�ner de cons) (1998)	4
430649	Dinner Game, The (Le D�ner de cons) (1998)	4
430650	Dinner Game, The (Le D�ner de cons) (1998)	4
430651	Dinner Game, The (Le D�ner de cons) (1998)	3
430652	Dinner Game, The (Le D�ner de cons) (1998)	1
430653	Dinner Game, The (Le D�ner de cons) (1998)	1
430654	Dinner Game, The (Le D�ner de cons) (1998)	4
430655	Dinner Game, The (Le D�ner de cons) (1998)	3
430656	Dinner Game, The (Le D�ner de cons) (1998)	4
430657	Dinner Game, The (Le D�ner de cons) (1998)	3

Feature Engineering

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 [33]:

```
x = master['Genre'].apply(lambda x : x.split('|')[0]).unique()
x
```

Out[33]:

```
array(['Animation', 'Adventure', 'Comedy', 'Action', 'Drama', 'Thril
ler',
       'Crime', 'Romance', "Children's", 'Documentary', 'Sci-Fi',
       'Horror', 'Western', 'Mystery', 'Film-Noir', 'War', 'Fantas
y',
       'Musical'], dtype=object)
```

Creating a new column name genre

In [34]:

```
master['genre'] = master['Genre'].apply(lambda x : x.split('|')[0])
```

In [35]:

```
master.shape
```

Out[35]:

```
(645833, 11)
```

In [36]:

```
master.head()
```

Out[36]:

	Id	Title	Genre	Gender	Age	X	occupation	Review	Rating	M
0	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	1193	5	97
1	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	661	3	97
2	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	914	3	97
3	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	3408	4	97
4	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	2355	5	97

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

In [81]:

```
x = master.values
```

Importing LAbelEncoder Class with onehotencoder

In [87]:

```
from sklearn.preprocessing import LabelEncoder , OneHotEncoder
## Creating an object for LabelEncoder
labelencoder_iv = LabelEncoder()

x[:,10]=labelencoder_iv.fit_transform(x[:,10])

onehotencoder = OneHotEncoder(categorical_features='all')
iv= onehotencoder.fit_transform(x).toarray()
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/_encode
rs.py:380: DeprecationWarning: The 'categorical_features' keyword is
deprecated in version 0.20 and will be removed in 0.22. The passed v
alue of 'all' is the default and can simply be removed.
  DeprecationWarning)
```

In [83]:

```
iv.shape
```

Out[83]:

```
(645833, 7587)
```

In [85]:

```
iv = pd.DataFrame(iv)
```

In [86]:

```
iv.head()
```

Out[86]:

	0	1	2	3	4	5	6	7	8	9	...	7577	7578	7579	7580	7581	7582
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	1.0	0.0	1.0	0.0
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	1.0	0.0	1.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	1.0	0.0	1.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	1.0	0.0	1.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	1.0	0.0	1.0	0.0

5 rows × 7587 columns

Linear Regression Model

Master dataset with sub dataset of Movies,Users and Rating

head() to view the top 5 row of the master dataset

In [40]:

```
master.head()
```

Out[40]:

	Id	Title	Genre	Gender	Age	X	occupation	Review	Rating	M
0	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	1193	5	97
1	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	661	3	97
2	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	914	3	97
3	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	3408	4	97
4	1	Toy Story (1995)	Animation|Children's|Comedy	F	1	10	48067	2355	5	97

Droping the columns which is less significane to the dataset

In [41]:

```
master.drop(labels = ['Title','Genre','MobileNo','occupation','X'],axis =1,inpla
ce = True)
```

Creating the dummies of Gender and genre in the dataset

In [42]:

```
dummy_cities=pd.get_dummies(master['Gender'])
master=pd.concat([master,dummy_cities],axis=1)
```

In [43]:

```
dummy_cities=pd.get_dummies(master['genre'])
master=pd.concat([master,dummy_cities],axis=1)
```

In [44]:

```
master.shape
```

Out[44]:

```
(645833, 26)
```

After creating the dummies droping the orginal column from the dataset

In [45]:

```
master.drop(labels = ['genre'],axis =1,inplace = True)
```

In [46]:

```
master.drop(labels = ['Gender'],axis =1,inplace = True)
```

In [47]:

```
# master.drop(labels = ['Age'],axis =1,inplace = True)
```

In [48]:

```
master.Id.nunique()
```

Out[48]:

```
3883
```

In [49]:

```
master.shape
```

Out[49]:

```
(645833, 24)
```

Master dataset view after creating the dummies

In [50]:

```
master.head()
```

Out[50]:

	Id	Age	Review	Rating	F	M	Action	Adventure	Animation	Children's	...	Fantasy
0	1	1	1193	5	1	0	0	0	1	0	...	0
1	1	1	661	3	1	0	0	0	1	0	...	0
2	1	1	914	3	1	0	0	0	1	0	...	0
3	1	1	3408	4	1	0	0	0	1	0	...	0
4	1	1	2355	5	1	0	0	0	1	0	...	0

5 rows × 24 columns

Dividing the Dataset into independent and dependent variables

In [51]:

```
dv = master[['Review']]
```

In [52]:

```
iv = master[['Id', 'Rating', 'F', 'M', 'Action', 'Adventure',
       'Animation', "Children's", 'Comedy', 'Crime', 'Documentary', 'Drama',
       'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance',
       'Sci-Fi', 'Thriller', 'War', 'Western']]
```

Splitting the master dataset into training and testing with ratio of 80:20 at random sample 0

In [53]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(iv,dv,test_size = 0.20,random_s
tate = 0)
```

After splitting the dataset shape of the training and testing

In [54]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(516666, 22)
(129167, 22)
(516666, 1)
(129167, 1)
```

In [55]:

```
X_train.head()
```

Out[55]:

	Id	Rating	F	M	Action	Adventure	Animation	Children's	Comedy	Crime	...
546426	3419	4	0	1	0	0	0	0	1	0	...
59805	408	5	0	1	0	0	0	0	0	0	...
245916	1509	2	0	1	0	0	0	0	0	0	...
575514	3576	3	0	1	1	0	0	0	0	0	...
293692	1797	5	0	1	0	0	0	0	0	0	...

5 rows × 22 columns

In [56]:

```
X_train = X_train.values
```

In [57]:

```
X_test  = X_test.values
```

In [58]:

```
y_train = y_train.values
```

Feature Scaling

In [59]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:,[0,1]] = sc.fit_transform(X_train[:,[0,1]])
X_test[:,[0,1]] = sc.fit_transform(X_test[:,[0,1]])

y_train[:,[0]] = sc.fit_transform(y_train[:,[0]])
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:5
95: DataConversionWarning: Data with input dtype int64 was converted
to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:5
95: DataConversionWarning: Data with input dtype int64 was converted
to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:5
95: DataConversionWarning: Data with input dtype int64 was converted
to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:5
95: DataConversionWarning: Data with input dtype int64 was converted
to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:5
95: DataConversionWarning: Data with input dtype int64 was converted
to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:5
95: DataConversionWarning: Data with input dtype int64 was converted
to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [60]:

```
X= pd.DataFrame(X_train)
X.head()
```

Out[60]:

	0	1	2	3	4	5	6	7	8	9	...	12	13	14	15	16	17	18	19	20	21
0	1	0	0	1	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
1	-1	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	-1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 22 columns

Linear Regression Model

In [61]:

```
from sklearn.linear_model import LinearRegression
x = LinearRegression()
model = x.fit(X_train,y_train)
```

In [62]:

```
model
```

Out[62]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Predicting the values on the independent test dataset i.e X_test

In [63]:

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

In [64]:

```
y_pred = pd.DataFrame(y_pred)
```

Interpretation of the Model

R Square of the model

The coefficient of the independent variables

The intercept of the model

In [65]:

```
print("The R Square of the model is " ,model.score(X_train,y_train))
print("The Coefficient if Regression is/are" ,model.coef_)
print("The Intercept of the model is " ,model.intercept_)
```

```
The R Square of the model is  0.0005935162897449109
The Coefficient if Regression is/are [[-1.79139975e-03 -1.34532086e-
02  3.27870531e+06  3.27870532e+06
  -3.21225042e+09 -3.21225042e+09 -3.21225042e+09 -3.21225042e+09
  -3.21225042e+09 -3.21225042e+09 -3.21225042e+09 -3.21225042e+09
  -3.21225042e+09 -3.21225042e+09 -3.21225042e+09 -3.21225042e+09
  -3.21225042e+09 -3.21225042e+09 -3.21225042e+09 -3.21225042e+09
  -3.21225042e+09 -3.21225042e+09]]
The Intercept of the model is  [3.20897171e+09]
```

Root square mean error

In [66]:

```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_test,y_pred))
```

In [67]:

```
# Root squared mean error
rms
```

Out[67]:

```
2178.579328893946
```

Graph for linear regression model

In [68]:

```
y_pred.columns = ['prediction']
print(y_pred.head())
print(y_pred.shape)
```

```
   prediction
0    0.009064
1    0.022167
2    0.002962
3   -0.006566
4    0.031790
(129167, 1)
```

In [69]:

```
df = pd.DataFrame()
```

In [70]:

```
y_pred.shape
```

Out[70]:

```
(129167, 1)
```

In [71]:

```
y_pred.head()
```

Out[71]:

	prediction
0	0.009064
1	0.022167
2	0.002962
3	-0.006566
4	0.031790

In [72]:

```
y_test = pd.DataFrame(y_test)
y_test.head()
```

Out[72]:

	Review
245690	3000
158780	2333
625481	919
466377	3176
590829	123

In [73]:

```
y_test.columns = ['Review']
```

In [74]:

```
y_test = y_test.reset_index()
y_test.head()
```

Out[74]:

	index	Review
0	245690	3000
1	158780	2333
2	625481	919
3	466377	3176
4	590829	123

In [75]:

```
print(y_test.shape)
print(y_test.head())
```

```
(129167, 2)
    index  Review
0  245690    3000
1  158780    2333
2  625481     919
3  466377    3176
4  590829     123
```

In [76]:

```
df = pd.concat([y_test,y_pred],axis =1)
df.head()
```

Out[76]:

	index	Review	prediction
0	245690	3000	0.009064
1	158780	2333	0.022167
2	625481	919	0.002962
3	466377	3176	-0.006566
4	590829	123	0.031790

In [78]:

```
df = df.drop(['index'],axis =1)
```

In [79]:

```
df.head()
```

Out[79]:

	Review	prediction
0	3000	0.009064
1	2333	0.022167
2	919	0.002962
3	3176	-0.006566
4	123	0.031790

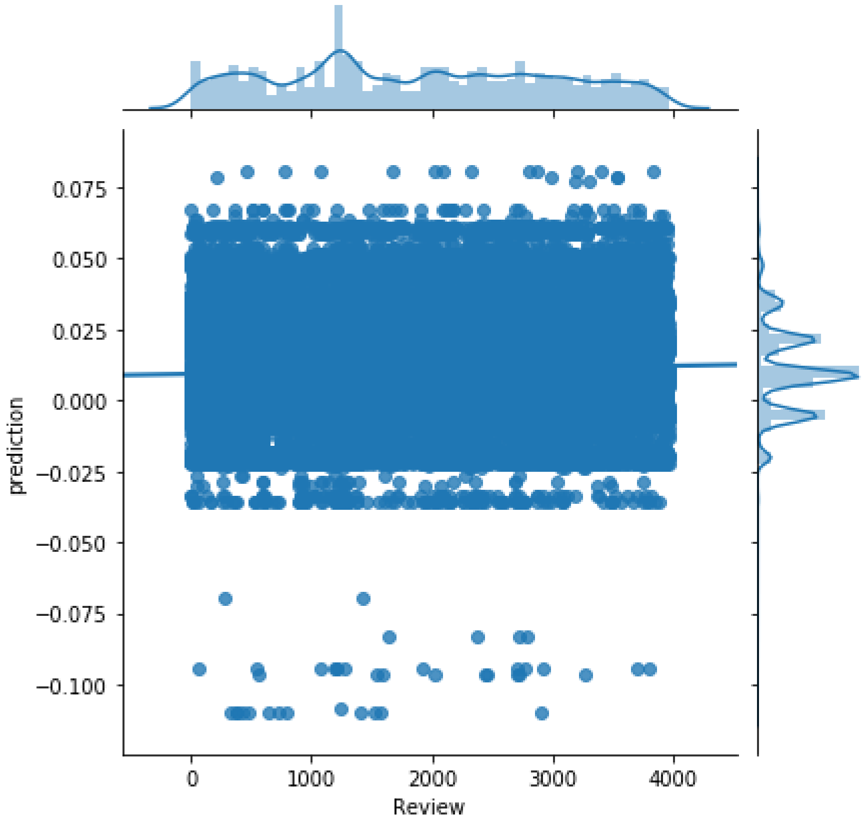
LInear Regression Plot

In [80]:

```
sns.jointplot('Review','prediction',data = df,kind ='reg')
```

Out[80]:

```
<seaborn.axisgrid.JointGrid at 0x1a30846438>
```



In [ ]:

```

```