**Machine Learning**

**Krishna Khatri**

**Part 1) Linear Regression**

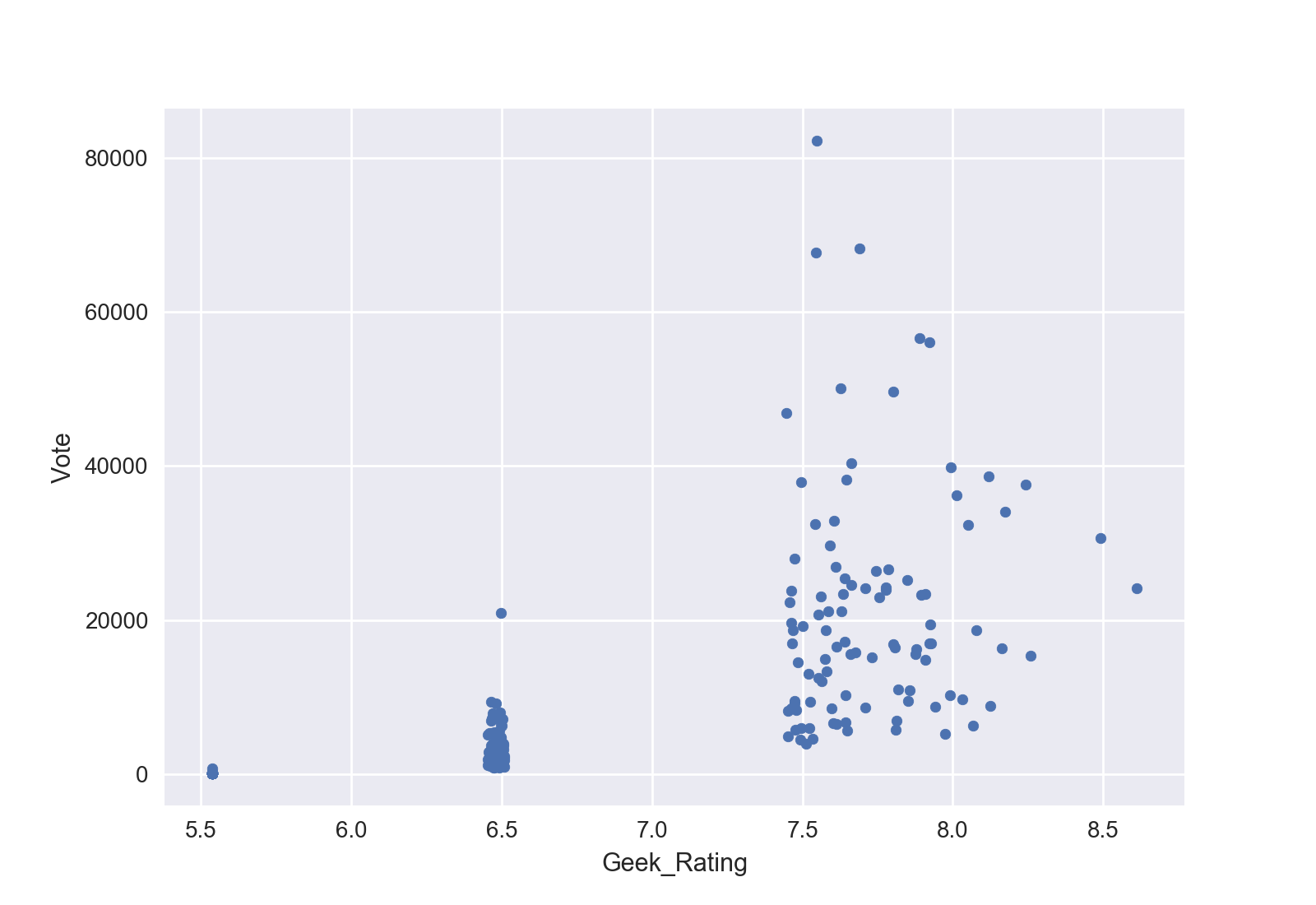
I have collected data by web scraping on Vote, Greek\_Rating and Avg\_Rating of different type of board games. My interest lies in identifying whether Greek\_Rating and Avg\_Rating help us to predict vote received by particular board game. I am using linear regression model and have imported statsmodels package. Data from first 200 types of board games are used for this machine learning analysis.

Import following package in text editor:

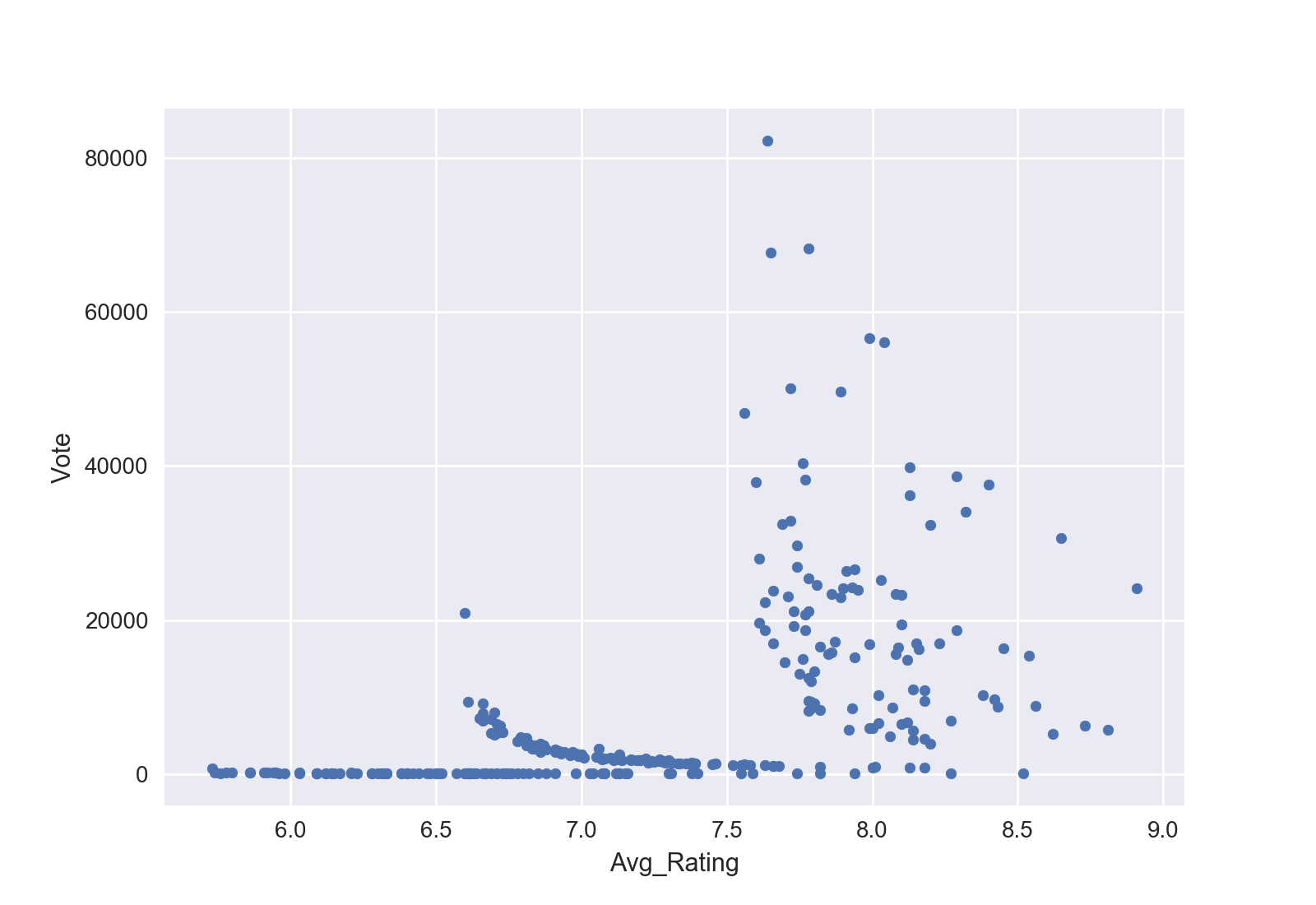
|  |  |
| --- | --- |
|  | import pandas as pd |
|  | import seaborn as sns |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | from sklearn.svm import SVC |
|  | from sklearn import svm |
|  | from sklearn.metrics import confusion\_matrix |
|  | from sklearn.metrics import classification\_report |
|  | from sklearn.preprocessing import StandardScaler, LabelEncoder |
|  | from sklearn.model\_selection import train\_test\_split |
|  | from sklearn.ensemble import RandomForestClassifier |
|  | from sklearn.cluster import KMeans |
|  | from sklearn.mixture import GaussianMixture |
|  | from sklearn import metrics |
|  | from sklearn.neural\_network import MLPClassifier |
|  | from sklearn import linear\_model |
|  | import statsmodels.api as sm |

Then, read the data using command call pd.read\_csv and save it to df1. Matplotlib.pyplot is imported as plt. Style of plt is set to be seaborn.

Initially, my hypothesis is that Geek\_Average is positively related to vote received by a particular type of game. However, I got negative result from scatter plot and OLS regression result. Scatter plot of Geek\_Rating and Vote is shown is following table:

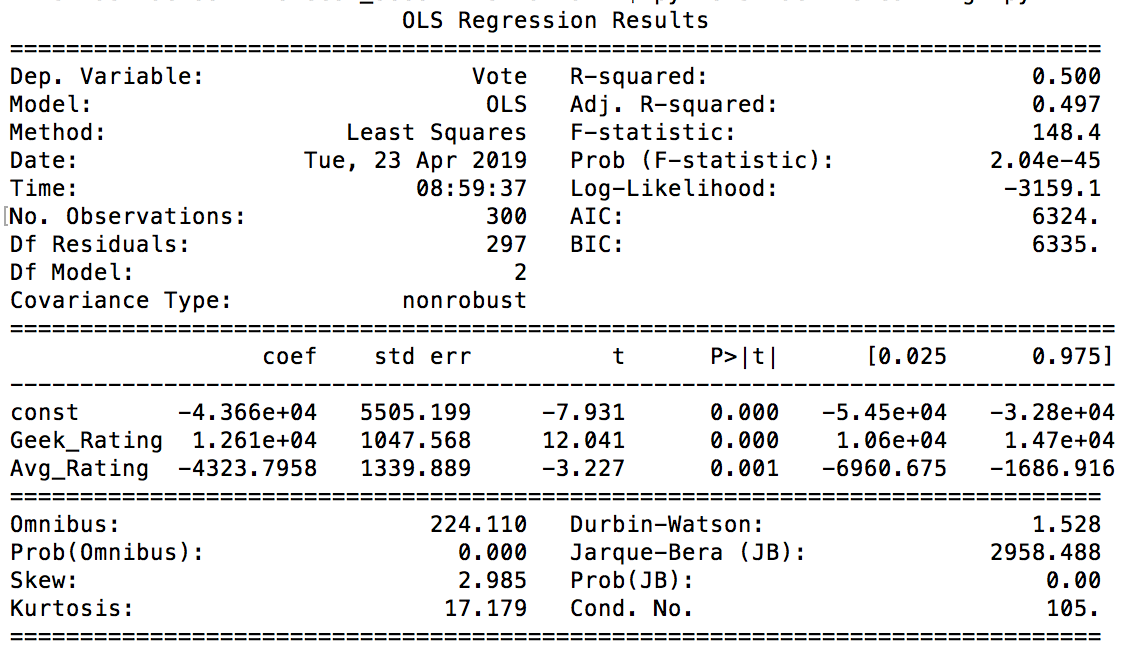


Also, my initial guess about the relationship of vote and Avg\_Rating was positive. Scatterplot and ols regression show that vote and Avd\_Rating are positively related. Scatter plot of Avg\_Rating and Vote is shown below.



OLS regression is run by using following command and result obtained from regression is pasted below.

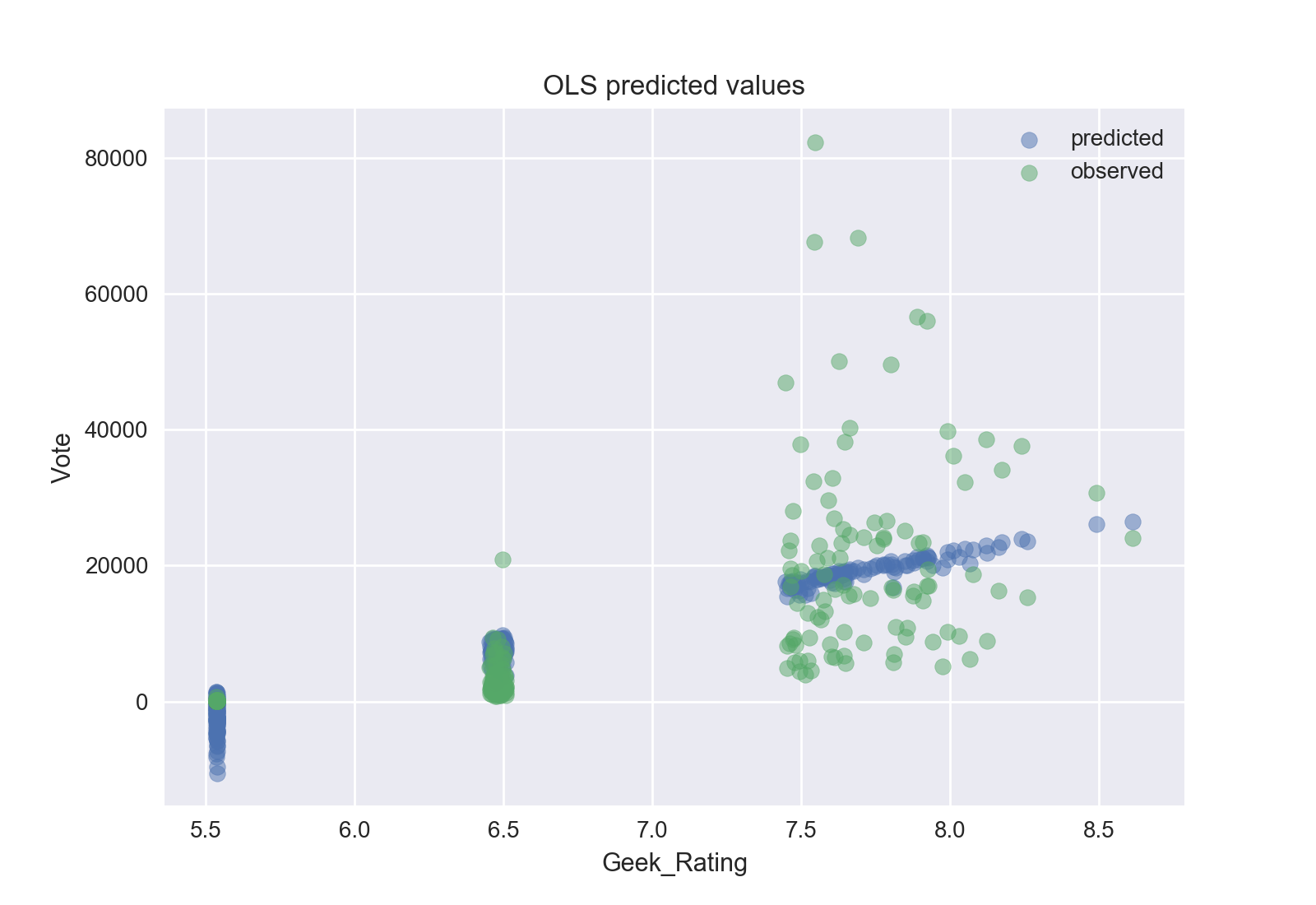
reg1 = sm.OLS(endog=df1['Vote'], exog=df1[['const','Geek\_Rating', 'Avg\_Rating']])

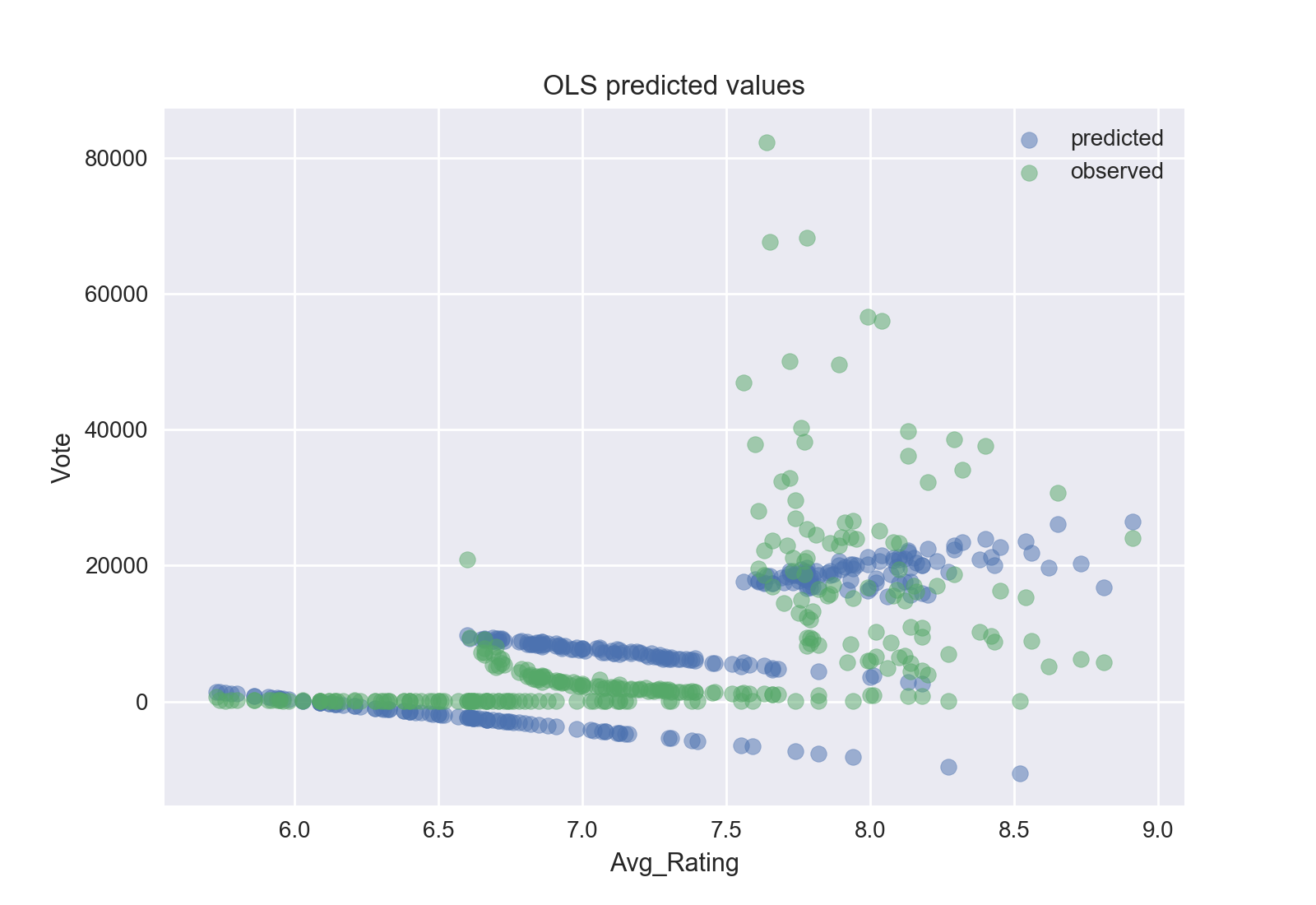


I found that Geek\_Rating is positively and Avg\_Rating is negatively related to Vote. One point increases in Geek\_Rating leads to increase in vote number by 12610 and one point increase in Avg\_Rating actually leads to reduction in Vote number by 4324.

Using OLS coefficients and mean value of independent variables, I predict the Vote received by particular board game using Geek\_Rating of board game and Avg\_Rating of board game.

Following figures summarize the prediction results.





**Part 2) Classification using sklearn**

Here, I am interested in classifying games by either receiving enough vote or receiving not enough vote based on their vote count. I am using data on vote, Geek\_Rating and Avg\_Rating of first 200 board games. Game with highest number of vote is 83000 and game with lowest number of vote is 30. So I need to rescale it.

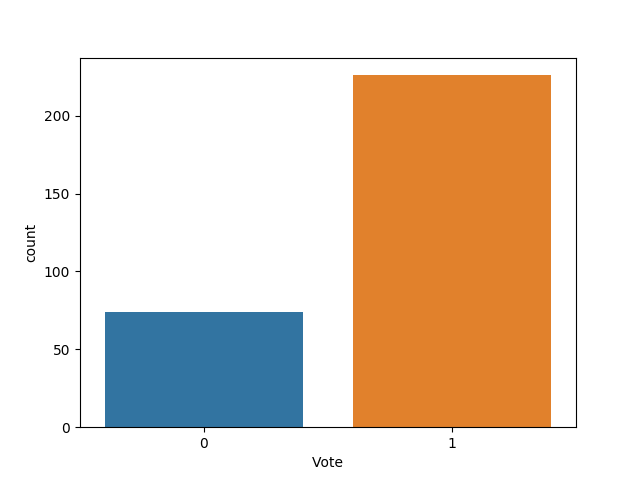
Import following commands in text editor:

|  |  |
| --- | --- |
|  | import seaborn as sns |
|  | import numpy as np  import pandas as pd |
|  | import matplotlib.pyplot as plt |
|  | from sklearn.svm import SVC |
|  | from sklearn import svm |
|  | from sklearn.metrics import confusion\_matrix |
|  | from sklearn.metrics import classification\_report |
|  | from sklearn.preprocessing import StandardScaler, LabelEncoder |
|  | from sklearn.model\_selection import train\_test\_split |
|  | from sklearn.ensemble import RandomForestClassifier |
|  | from sklearn.cluster import KMeans |
|  | from sklearn.mixture import GaussianMixture |
|  | from sklearn import metrics |
|  | from sklearn.neural\_network import MLPClassifier |

Read the data using pd.read\_csv command and save it to df. Cut off number of vote is set to be 10000 so anything less than that implies game with not enough vote and game with more than 10000 vote are classified as game with enough vote. After using labelencoder and label\_ratings.fit\_transformation command to Vote, I plot figure of enough vote and not enough vote.

Then training set is set as X\_train, y\_train and test set is set as X\_test and y\_test. After scaling data, I use 3 different types of classifier to train and test using prediction. I summarize that in confusion matrix table.

Classifiers that I am using are RandomForestClassifier(rfc), svm.SVC and MLPClassifier(mlpc). Comparison between confusion matrix of above mentioned classifiers shows that MLPClassifier is better in predicting than other two classifiers in current situations.



First five heading:

Avg\_Rating ... Vote

0 8.91 ... 24101

1 8.65 ... 30682

2 8.54 ... 15375

3 8.40 ... 37613

4 8.32 ... 34064

Data Information:

RangeIndex: 300 entries, 0 to 299

Data columns (total 4 columns):

Avg\_Rating 300 non-null float64

Geek\_Rating 300 non-null float64

Title 300 non-null object

Vote 300 non-null int64

Description of Data:

Avg\_Rating Geek\_Rating Vote

count 300.000000 300.000000 300.000000

mean 7.229567 6.582053 8101.736667

std 0.711221 0.909685 12830.171937

min 5.730000 5.537000 30.000000

25% 6.700000 5.538000 89.000000

50% 7.170000 6.477000 2270.500000

75% 7.792500 7.542750 9508.500000

max 8.910000 8.612000 82269.000000

Avg\_Rating Geek\_Rating Title Vote

0 8.91 8.612 Gloomhaven 0

1 8.65 8.492 Pandemic Legacy: Season 1 0

2 8.54 8.258 Through the Ages: A New Story of Civilization 0

3 8.40 8.241 Terraforming Mars 0

4 8.32 8.175 Twilight Struggle 0

1 226 --🡪 Not Enough Vote

0 74 🡪 Enough Vote

Categories (2, object): [Not Enough < Enough]

Name: Vote, dtype: int64

[[-0.22514606 -1.1589296 ]

[ 1.21353196 1.14287245]

[ 1.86234754 1.71083517]

[ 0.59292576 -0.13393437]

[ 0.29672734 -0.12838786]

[ 1.26995071 1.3691701 ]

[ 1.24174134 1.4712259 ]

[-1.08553193 -1.1600389 ]

[ 0.762182 1.16062128]

[-0.38029761 -0.13393437]]

**Confusion Matrix: RandomForestClassifier**

precision recall f1-score support

0 0.78 0.93 0.85 15

1 0.98 0.91 0.94 45

micro avg 0.92 0.92 0.92 60

macro avg 0.88 0.92 0.90 60

weighted avg 0.93 0.92 0.92 60

[[14 1]

[ 4 41]]

**Confusion Matrix: Classifier**

precision recall f1-score support

0 0.75 1.00 0.86 15

1 1.00 0.89 0.94 45

micro avg 0.92 0.92 0.92 60

macro avg 0.88 0.94 0.90 60

weighted avg 0.94 0.92 0.92 60

[[15 0]

[ 5 40]]

Confusion Matrix: Neural Network Classifier

precision recall f1-score support

0 0.79 1.00 0.88 15

1 1.00 0.91 0.95 45

micro avg 0.93 0.93 0.93 60

macro avg 0.89 0.96 0.92 60

weighted avg 0.95 0.93 0.94 60

[[15 0]

[ 4 41]]