hw5_ds

December 7, 2022

[24]:
```
# the usual libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# statsmodels
import statsmodels.formula.api as smf

# SciKit Learn libraries
from sklearn.model_selection import train_test_split # for splitting data
from sklearn.linear_model import LinearRegression    # linear regression
from sklearn.preprocessing import StandardScaler     # feature scaling
from sklearn import metrics                          # for evaluation metrics
from sklearn.neighbors import KNeighborsClassifier   # knn
```

[25]:
```
#Q1(a)
data=pd.read_csv("bechdel.csv")
data.head(5)
```

[25]:

	year	title	bechdel	budget	domgross	intgross	rated	\
0	2013	21 & Over	FAIL	13.000000	25.682380	42.195766	other	
1	2012	Dredd 3D	PASS	45.658735	13.611086	41.467257	other	
2	2013	12 Years a Slave	FAIL	20.000000	53.107035	158.607035	R	
3	2013	2 Guns	FAIL	61.000000	75.612460	132.493015	R	
4	2013	42	FAIL	40.000000	95.020213	95.020213	PG-13	

	imdb_rating	romcom	drama	action	sci_fi	runtime
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN
2	8.3	0.0	1.0	0.0	0.0	134.0
3	6.8	0.0	0.0	1.0	0.0	109.0
4	7.6	0.0	1.0	0.0	0.0	128.0

Q1(b)Movies is the unit of analysis

[26]:
```
##Q1(c):
print("The number of features in this dataset are", len(data.columns))
print("The number of observations in this dataset are",len(data.index))
```

```
The number of features in this dataset are 13
The number of observations in this dataset are 1794
```

[27]:
```
##Q1(d):
data['rated'].value_counts()
```

[27]:
```
R        691
PG-13    565
other    281
PG       257
Name: rated, dtype: int64
```

Q1e: The dataset is not a good representation of movies, as the Bechdel test contributions come from voluntary people contribution, which is why it might not be the best representation.

[28]:
```
#Q1(f):
data_set=data.dropna(subset=["runtime"])
data_set.shape
```

[28]:
```
(1591, 13)
```

[29]:
```
#Q2(a)
model = smf.ols(formula='imdb_rating ~ budget+ drama+sci_fi+romcom',␣
 ↪data=data_set).fit()
model.summary()
```

[29]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            imdb_rating   R-squared:                       0.079
Model:                            OLS   Adj. R-squared:                  0.077
Method:                 Least Squares   F-statistic:                     34.04
Date:                Wed, 07 Dec 2022   Prob (F-statistic):           2.77e-27
Time:                        22:03:22   Log-Likelihood:                -2131.6
No. Observations:                1591   AIC:                             4273.
Df Residuals:                    1586   BIC:                             4300.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      6.4645      0.046    140.073      0.000       6.374       6.555
budget         0.0012      0.000      2.635      0.009       0.000       0.002
drama          0.5445      0.049     11.198      0.000       0.449       0.640
sci_fi        -0.0378      0.071     -0.530      0.596      -0.178       0.102
romcom        -0.2111      0.086     -2.469      0.014      -0.379      -0.043
==============================================================================
```

```
Omnibus:                       82.261   Durbin-Watson:                   1.857
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              109.843
Skew:                          -0.484   Prob(JB):                     1.41e-24
Kurtosis:                       3.849   Cond. No.                         298.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Q2(b) the variance shown is 7.7% which is the value of adjusted r-squared.

Q2(c) Romcom,budget and drama are significant at the 5% level.

Q2(d): It is associated with a 0.2111 decrease in imdb_rating.

[30]:
```
##Q2(e)
imdb_rating=0.0012*100
imdb_rating
```

[30]:
```
0.12
```

Q3(A): Since the value of adjusted r-squared is 7.7%,which is quite low, it will do a bad job explaining the data.

[31]:
```
#Q3(b)
x_df=data_set[['budget','drama','romcom','sci_fi']]
x_df.head(5)
```

[31]:

	budget	drama	romcom	sci_fi
2	20.0	1.0	0.0	0.0
3	61.0	0.0	0.0	0.0
4	40.0	1.0	0.0	0.0
5	225.0	0.0	0.0	0.0
6	92.0	0.0	0.0	0.0

[32]:
```
#Q3(b)
y_df=data_set[['imdb_rating']]
y_df.head(5)
```

[32]:

	imdb_rating
2	8.3
3	6.8
4	7.6
5	6.6
6	5.4

[33]:
```
#Q3(c)
x_df_train,x_df_test,y_df_train, y_df_test = train_test_split(x_df,y_df␣
 ↪,random_state=135, test_size=0.20, shuffle=True)
print(x_df_train.head(5))
print("number of values in x training set are ", len(x_df_train.index))
print("number of values in x testing set are ", len(x_df_test.index))

##https://www.geeksforgeeks.org/
 ↪how-to-do-train-test-split-using-sklearn-in-python/. Used these values to␣
 ↪determine how to to do train test split using Sklearn
```

```
          budget  drama  romcom  sci_fi
570    40.043484    1.0     0.0     0.0
957    27.130243    0.0     0.0     0.0
692    28.088587    0.0     0.0     0.0
114   101.463857    1.0     0.0     0.0
1752   53.560590    1.0     0.0     0.0
number of values in x training set are  1272
number of values in x testing set are  319
```

[34]:
```
#Q3(d)
regression = LinearRegression()
regression.fit(x_df_train, y_df_train)
print("the intercept is",regression.intercept_)
print("the coeffeicients are",regression.coef_)

##https://stackabuse.com/linear-regression-in-python-with-scikit-learn/, used␣
 ↪this website to check for the intercepts and coeffficients.
```

```
the intercept is [6.42565479]
the coeffeicients are [[ 0.00161455  0.58995507 -0.23433976 -0.0324223 ]]
```

Q3(e): The value of the intercept decreases a bit which means the y-intercept when using Sklearn is lower. Furthermore, the coeffiecients for drama increased, which means 1 unit change in iMDB_Rating will increase the value of drama by a bit more

[35]:
```
#Q3(f)
prediction=pd.DataFrame(regression.predict(x_df_test))
prediction.head(10)
```

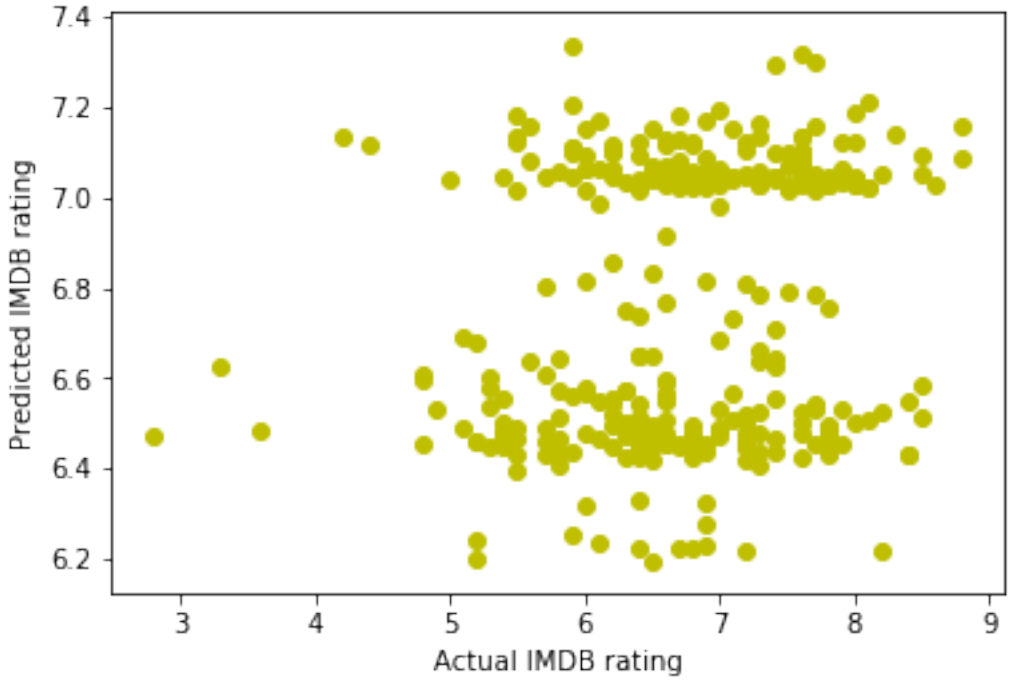
[35]:

	0
0	6.429693
1	6.473947
2	7.021981
3	6.487009
4	6.456654
5	7.044672
6	6.450228

```
7  6.530496
8  6.460149
9  6.450185
```

[36]:
```
#Q3(g)
plt.scatter(y_df_test,prediction,c='y')
plt.xlabel("Actual IMDB rating")
plt.ylabel("Predicted IMDB rating")

plt.show()
```



[37]:
```
##Q3(h)
import math
mse = metrics.mean_squared_error(y_df_test, prediction)

rmse = math.sqrt(mse)

print("the Root Mean Squared Error value is ",rmse)

##The RMSE value of 0.9284 tells us that there is an average difference of 0.
  ↪92847 between the actual value and the value predicted by our model.
```

```
##https://www.folkstalk.com/2022/10/
↪how-to-calculate-rmse-in-linear-regression-python-with-code-examples.html.␣
↪Use this website to learn how to calculate RMSE value.
```

```
the Root Mean Squared Error value is  0.9284746856773564
```

Q3(i): As the data shows, that the majority of IMDB rating are between 6 and 8, while our scatterplot and RMSE, show that there is a difference of 0.9285 approximately between predicted and actual values, which in such a short range, is a substantial difference and it can be said that the model is doing a poor job.

[38]:
```
#Q4(a)
data_set['bechdel'].value_counts()
```

[38]:
```
FAIL    892
PASS    699
Name: bechdel, dtype: int64
```

[39]:
```
#Q4(b)
count = data_set['bechdel'].count()
fail_bechdel = (892/count)*100
print("the percentage of movies failing bechdel test is", fail_bechdel)
```

```
the percentage of movies failing bechdel test is 56.06536769327467
```

[40]:
```
#Q4(c)
x_dataset=data_set[['year','budget','domgross','intgross','imdb_rating','romcom','drama','acti
x_dataset.head(5)
```

[40]:

	year	budget	domgross	intgross	imdb_rating	romcom	drama	action	\
2	2013	20.0	53.107035	158.607035	8.3	0.0	1.0	0.0	
3	2013	61.0	75.612460	132.493015	6.8	0.0	0.0	1.0	
4	2013	40.0	95.020213	95.020213	7.6	0.0	1.0	0.0	
5	2013	225.0	38.362475	145.803842	6.6	0.0	0.0	1.0	
6	2013	92.0	67.349198	304.249198	5.4	0.0	0.0	1.0	

	sci_fi
2	0.0
3	0.0
4	0.0
5	0.0
6	0.0

[41]:
```
#Q4(d)
x_data = x_dataset.reset_index()
x_dataframe = x_data
average_year = (x_data['year']).mean()
standard_deviation = (np.std(x_data['year']))
```

```
def mean(column):
    return (x_data[column].mean())
def std(column):
    return (np.std(x_data[column]))
for i in ['year','budget', 'domgross', 'intgross', 'imdb_rating']:
    average_year = mean(i)
    standard_deviation = std(i)
    for j in range (1591):
        x_dataframe.loc[j,i] = (x_data.loc[j,i] - average_year)/
 ↪standard_deviation
##dropping the index column.
x_dataframe = x_dataframe.drop(['index'], axis = 1)
x_dataframe.head(5)

# https://sparkbyexamples.com/pandas/normalize-columns-of-pandas-dataframe/,␣
 ↪https://www.digitalocean.com/community/tutorials/
 ↪normalize-data-in-pythonused these two websites to study the approch on how␣
 ↪to normalize and then used it to write code for myself.
```

[41]:

	year	budget	domgross	intgross	imdb_rating	romcom	drama	action	sci_fi
0	1.16439	-0.656777	-0.347975	-0.140206	1.599228	0.0	1.0	0.0	0.0
1	1.16439	0.089670	-0.160641	-0.234213	0.041131	0.0	0.0	1.0	0.0
2	1.16439	-0.292657	0.000907	-0.369110	0.872116	0.0	1.0	0.0	0.0
3	1.16439	3.075461	-0.470707	-0.186296	-0.166615	0.0	0.0	1.0	0.0
4	1.16439	0.654058	-0.229424	0.384084	-1.413092	0.0	0.0	1.0	0.0

[42]:
```
#Q4(e)
x_sklearn_train,x_sklearn_test,y_train, y_test =␣
 ↪train_test_split(x_dataframe,data_set['bechdel'] ,random_state=321,␣
 ↪test_size=0.20, shuffle=True)
x_sklearn_train.head(5)
```

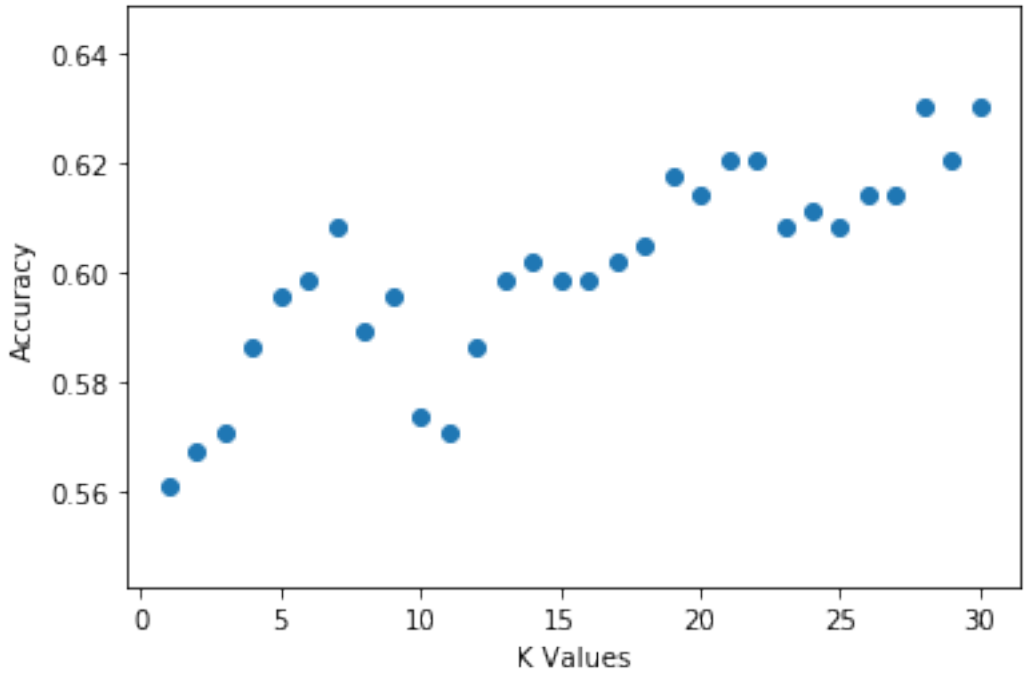
[42]:

	year	budget	domgross	intgross	imdb_rating	romcom	drama	action	sci_fi
1064	-0.165108	1.590343	2.666125	3.491451	2.222466	0.0	0.0	0.0	0.0
1415	-1.273023	-0.584831	-0.295460	-0.497280	-0.166615	0.0	1.0	0.0	0.0
512	0.610433	-0.429790	-0.220937	-0.301418	0.560497	1.0	1.0	0.0	0.0
1416	-1.383814	0.277539	0.511057	0.853008	0.664370	0.0	0.0	0.0	0.0
1375	-1.051440	0.887274	0.338073	0.769008	-0.478234	0.0	0.0	0.0	0.0

```
1064     1.0     0.0
1415     0.0     0.0
512      0.0     0.0
1416     0.0     1.0
1375     1.0     0.0
```

[43]:
```
#Q4(f)
from sklearn.metrics import accuracy_score
lst=[]
for j in range(1,31):
    lst.append(j)
lst_1=[]
for i in range(1,31):
    knn = KNeighborsClassifier(n_neighbors=i)
##Predicting results using Test data set
    knn.fit(x_sklearn_train,y_train)
    pred = knn.predict(x_sklearn_test)
    accuracy=accuracy_score(pred,y_test)
    lst_1.append(accuracy)
plt.scatter(lst,lst_1)
plt.xlabel("K Values")
plt.ylabel("Accuracy")
plt.show()

#https://towardsdatascience.com/k-nearest-neighbors-94395f445221, used this to␣
 ↪predict results using the test data set and then created two lists to plot␣
 ↪it.
```



[44]:
```
#Q4(g)
knn = KNeighborsClassifier(n_neighbors=28)

##Training the model using the training sets
knn.fit(x_sklearn_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(x_sklearn_test)
test=pd.DataFrame(y_pred,columns=['predictions'])
test.head(5)

##This value of K was chosen because it produced the highest accuracy, which I␣
 ↪gave preference to while conduction this experiment.
```

[44]:

	predictions
0	PASS
1	PASS
2	FAIL
3	PASS
4	FAIL

[45]:
```
#Q4(h)
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,test))

#75 passed the test that were supposed to fail the test.


##https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
 ↪confusion_matrix.html, use this website to figure out how to find confusion␣
 ↪matrix.
```

```
[[138  43]
 [ 75  63]]
```

[46]:
```
#Q4(i)
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
# The 0.46 value of recall means that out of all the testing, only 46% of␣
 ↪passses, were actually supposed to pass.

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
 ↪classification_report.html, used this for classification report.
```

```
              precision    recall  f1-score   support
```

```
        FAIL       0.65      0.76      0.70       181
        PASS       0.59      0.46      0.52       138

    accuracy                           0.63       319
   macro avg       0.62      0.61      0.61       319
weighted avg       0.62      0.63      0.62       319
```

Q4(j) The model does a poor job predicting if a movie passes the bechdel test or no because it produced a lot of wrong predictions as in 4h it can be seen, 75 filns passed the test that were supposed to fail. Similarly, the precision values are very low, and the model can not be declared to be accurate and a good predictor.