HW 5 olivia lewandowski

January 25, 2023

0.1 DS4E: Homework 5

[23]:
```
# 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
from sklearn.metrics import classification_report, confusion_matrix
```

0.2 Question 1

1(a)

[24]:
```
bechdel_data = pd.read_csv('bechdel.csv')
bechdel_data.head()
```

[24]:
```
   year             title bechdel     budget   domgross    intgross  rated  \
0  2013     21 &amp; 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
```

1(b)

The unit of analysis in this dataset is the movie. movies

1(c)

[25]:
```python
bechdel_data.shape
print("There are 1794 columns and 13 features")
```

```
There are 1794 columns and 13 features
```

1(d)

[26]:
```python
bechdel_data['rated'].value_counts()
```

[26]:
```
R         691
PG-13     565
other     281
PG        257
Name: rated, dtype: int64
```

1(e)

I do not think that this data is representative of the population of movies; first of all, this sample of movies was not acquired using random sampling, and therefore leaves room for potential bias within the data set. To get the data they relied on BechdelTest.com, which relies on voluntary contributions; this may allow for selection bias to occur, as volunteered movies may be of a certain quality and not fully representative of the whole population of movies. On top of that, the movies must also appear on The-Numbers.com, which may be restricted to limited movies or a certain type of movies, further contributing to the data not being representative of the population.

1(f)

[27]:
```python
bechdel_data = bechdel_data.dropna(subset = ['runtime'])
bechdel_data.head()
```

[27]:
```
   year                   title bechdel  budget   domgross    intgross  rated  \
2  2013        12 Years a Slave    FAIL    20.0  53.107035  158.607035      R
3  2013                  2 Guns    FAIL    61.0  75.612460  132.493015      R
4  2013                      42    FAIL    40.0  95.020213   95.020213  PG-13
5  2013                47 Ronin    FAIL   225.0  38.362475  145.803842  PG-13
6  2013  A Good Day to Die Hard    FAIL    92.0  67.349198  304.249198      R

   imdb_rating  romcom  drama  action  sci_fi  runtime
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
5          6.6     0.0    0.0     1.0     0.0    118.0
6          5.4     0.0    0.0     1.0     0.0     98.0
```

[28]:
```python
bechdel_data.shape
```

[28]: (1591, 13)

0.3 Question 2

2(a)

[29]:
```
bechdel_regression = smf.ols('imdb_rating ~ budget + drama + sci_fi + romcom',␣
↪data = bechdel_data).fit()
bechdel_regression.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:                Thu, 08 Dec 2022   Prob (F-statistic):           2.77e-27
Time:                        14:04:52   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.
"""
```

2(b)

7.7% of the variance in imdb rating is explained in our features; this can be seen in the adjusted r-squared value in our regression summary, which is 0.077. We used the adjusted r-squared because it adjusts for added variables; if we didn't adjust, simply adding variables will push up r-squared.

2(c)

The features significant at the 0.05 p-value, or 5% level: budget, drama, and romcom. This can be observed in the regression summary, as budget's p-value is 0.009, drama's p-value is 0.000, and romcom's p-value is 0.014, which are all under the 0.05 p-value or 5% level. However, when observing the confidence interval, romcom and drama's confidence interval don't contain zero, but budget's does, which is an indicator that it is not statistically significant.

2(d)

Holding budget constant, a 1-unit increase in romcom is associated with a -0.2111 change in imdb rating.

2(e)

Increasing the budget for a movie by 100 million dollars would change the imdb rating by (0.0012 x 100 = 0.12) percent. In other words, it would increase the imdb rating by 0.12 with a $100 million increase in budget.

0.4 Question 3

3(a)

Based on the statsmodel output from Q2, I expect that the four features will do a bad job at predicting IMDB ratings; firstly, the correlation values are relatively low, but the main indicator that the model won't be a good fit for the data is the r-squared value. The r-squared value is extremely low, at only a 0.077, while most models need at least a 0.5 to even be considered truly, showing that the model is not a good fit for the data, and it won't predict IMDB ratings very well.

3(b)

[30]:
```
x_dataframe = bechdel_data[['budget', 'drama', 'romcom', 'sci_fi']]
x_dataframe.head()
```

[30]:

	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

[31]:
```
y_dataframe = bechdel_data[['imdb_rating']]
y_dataframe.head()
```

[31]:

	imdb_rating
2	8.3
3	6.8
4	7.6
5	6.6
6	5.4

3(c)

[32]:
```python
x_train, x_test, y_train, y_test = train_test_split(x_dataframe, y_dataframe,␣
↪test_size = 0.2, random_state = 135)
x_train.head()
```

[32]:

	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

[33]:
```python
print("Number of observations in X Train: " + str(len(x_train)))
```

```
Number of observations in X Train: 1272
```

[34]:
```python
print("Number of observations in X Test: " + str(len(x_test)))
```

```
Number of observations in X Test: 319
```

3(d)

[35]:
```python
regressor = LinearRegression()
regressor.fit(x_train, y_train)

print('Intercept', regressor.intercept_)
print('Coefficients', regressor.coef_)
```

```
Intercept [6.42565479]
Coefficients [[ 0.00161455  0.58995507 -0.23433976 -0.0324223 ]]
```

3(e)

In Q2, the estimated coefficient for drama was 0.5445, while the estimated coefficient in Q3 is 0.58995507. This coefficient is slighty more than the other, meaning there would be a slight increase in correlation between statsmodels and skikit/training data. The coefficients are not the same, but pretty similar considering Q3 is only using training data.

3(f)

[36]:
```python
y_pred = regressor.predict(x_test)

first_ten_data = pd.DataFrame(y_pred)
first_ten_data.head(10)
```

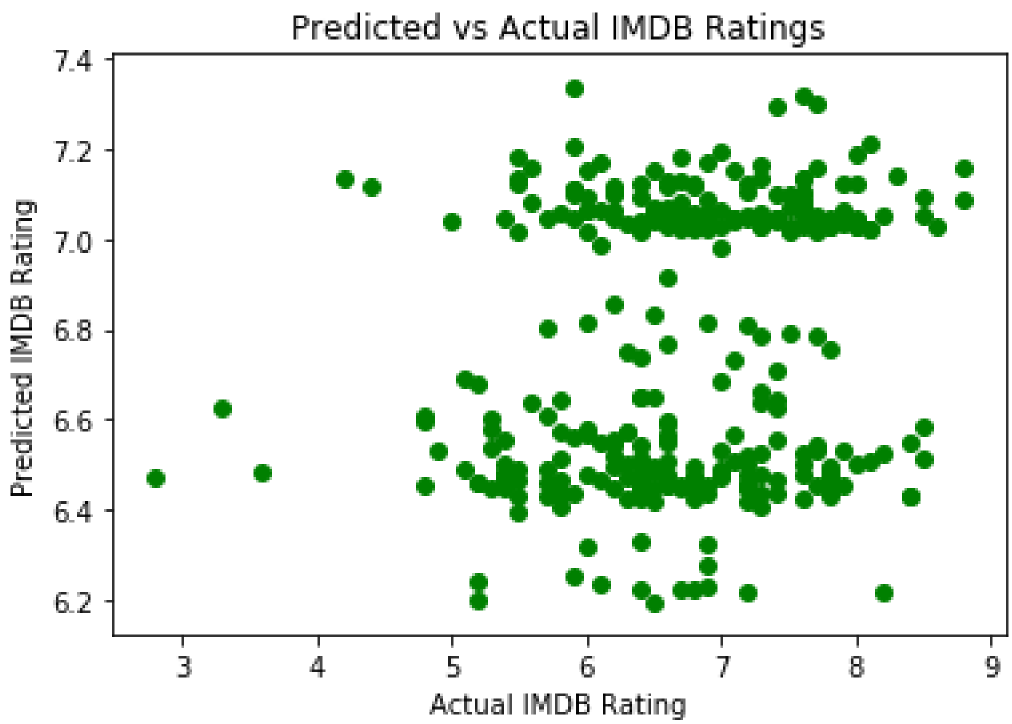
[36]:

	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
```

3(g)

```
[37]: plt.scatter(y_test, y_pred, color = 'green')
      plt.title('Predicted vs Actual IMDB Ratings')
      plt.xlabel('Actual IMDB Rating')
      plt.ylabel('Predicted IMDB Rating')
      plt.show()
```



3(h)

```
[38]: print('Root Mean Squared Error:',
            np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Root Mean Squared Error: 0.9284746856773564
```

Root mean squared error is the square root of the mean squared residual, wiht a high penalty on errors. It shows how far prediction values vary from actual measured values; in terms of average error, the higher the RMSE, the higher indication of average error in the prediction model, and the worse the model is.

3(i)

Reflecting on my analyses, I feel like this model does a poor job of predicting IMDB movie ratings. First of all, the RMSE is high; it is on a scale of 0 to 1, and the RMSE is 0.9285, which is on the higher side, showing how there is a high amount of average error and that our model doesn't fit the data set very well. Second, the scatterplot confirms this, showing how the predicted ratings hover around two spots, while the actual ratings are spread out more evenly; there is no linearity in the scatterplot, showing that the model is not a good fit.

0.5 Question 4

4(a)

[39]:
```
bechdel_data['bechdel'].value_counts()
```

[39]:
```
FAIL    892
PASS    699
Name: bechdel, dtype: int64
```

4(b)

If we built a classifier that always guessed that a movie failed the Bechdel test, the classifier would be correct 892/1,591 or 56% of the time.

4(c)

[40]:
```
classifier_data = bechdel_data[['year', 'budget', 'domgross', 'intgross',
'imdb_rating', 'romcom', 'drama', 'action', 'sci_fi']]
classifier_data.head()
```

[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
```

4(d)

[41]:
```
cols = ['year', 'budget', 'domgross', 'intgross', 'imdb_rating']
x_features = bechdel_data[cols]
scaler = StandardScaler()
scaled_x_features = scaler.fit_transform(x_features)
```

```
pd.DataFrame(scaled_x_features, columns = cols).head()
```

[41]:

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

4(e)

[42]:
```
y_data = bechdel_data[['bechdel']]
x_train, x_test, y_train, y_test = train_test_split(scaled_x_features, y_data,
    test_size = 0.2, random_state = 321)
pd.DataFrame(x_train, columns = cols).head(3)
```

[42]:

	year	budget	domgross	intgross	imdb_rating
0	-0.165108	1.590343	2.666125	3.491451	2.222466
1	-1.273023	-0.584831	-0.295460	-0.497280	-0.166615
2	0.610433	-0.429790	-0.220937	-0.301418	0.560497

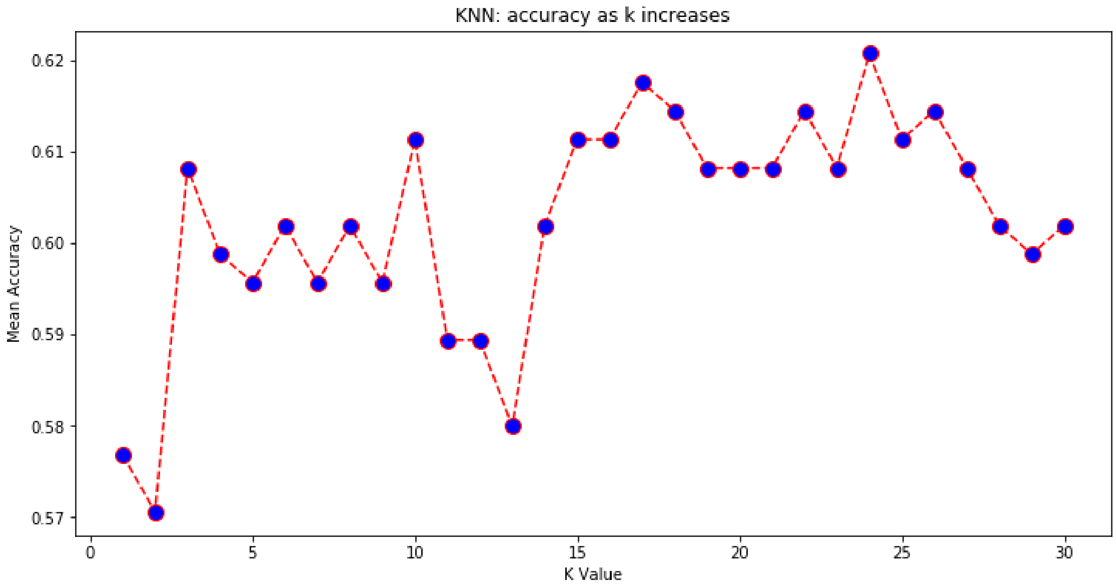
4(f)

[43]:
```
accuracy = list()

for i in range(1, 31):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train.to_numpy().flatten())
    pred_i = knn.predict(x_test)
    accuracy.append(metrics.accuracy_score(y_test, pred_i) )

plt.figure(figsize = (12, 6))
plt.plot(range(1, len(accuracy)+1), accuracy, color='red', linestyle='dashed',
    marker='o',
         markerfacecolor='blue', markersize=10)

plt.title('KNN: accuracy as k increases')
plt.xlabel('K Value')
plt.ylabel('Mean Accuracy')
plt.show()
```



4(g)

[44]:
```
new_classifier = KNeighborsClassifier(n_neighbors = 7)
new_classifier.fit(x_train, y_train.values.flatten())
y_pred = new_classifier.predict(x_test)
pd.DataFrame(y_pred).head(5)
```

[44]:

	0
0	FAIL
1	PASS
2	PASS
3	PASS
4	FAIL

I chose 7 as the k value for the KNN classifier because k is usually small (single digit), and if the k value is too high it can make the model too specific and unrealistic to real data. I also chose an odd number because this is usually better (because it splits the tie if necessary). Although the plot shows that the higher values in the 20s have a higher accuracy, they don't make sense in the context of the real world because it smaller numbers are used more.

4(h)

[45]:
```
cmat = confusion_matrix(y_test, y_pred)

print('TP - True Pass: {}'.format(cmat[0,0]))
print('FP - False Pass: {}'.format(cmat[0,1]))
print('FF - False Fail: {}'.format(cmat[1,0]))
print('TF - True Fail: {}'.format(cmat[1,1]))
```

```
print('Accuracy Rate: {}'.format(np.divide(np.sum([cmat[0,0],cmat[1,1]]),np.
 ↪sum(cmat))))
print('Misclassification Rate: {}'.format(np.divide(np.
 ↪sum([cmat[0,1],cmat[1,0]]),np.sum(cmat))))
```

```
TP - True Pass: 119
FP - False Pass: 62
FF - False Fail: 67
TF - True Fail: 71
Accuracy Rate: 0.5956112852664577
Misclassification Rate: 0.4043887147335423
```

Of 119 movies that passed the bechdel test (True Pass), 71 were predicted to have failed the Bechdel test.

4(i)

[46]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[119  62]
 [ 67  71]]
              precision    recall  f1-score   support

        FAIL       0.64      0.66      0.65       181
        PASS       0.53      0.51      0.52       138

    accuracy                           0.60       319
   macro avg       0.59      0.59      0.59       319
weighted avg       0.59      0.60      0.59       319
```

Recall shows the ability of a classifier/model to correctly find the positive instances/relevant cases in the data (ratio of true positives vs false negatives). The recall for the classification "PASS" is 0.51, suggests about our model that it did not do a good job of finding the positive instances of PASS, as it is almost at 0.50 which is what it would be if the prediction was completely random (half and half).

4(j)

Reflecting on the analysis above, do you feel like this model does a good or poor job of predicting whether a movie passes or fails the Bechdel test?

I feel like this model does a poor job of predicting whether a movie passes or fails the bechdel test; all of the values in the classification report are close to 50%, which is what it would be by default if the prediction was random. Furthermore, the accuracy rate is only 59%, which again isn't great considering the default would be 50%, and the misclassification rate is 41%, which correlates in that 41% of the time it gets the answer wrong.