km5188_ds4e_hw5

December 8, 2022

1 DS4E: Homework 4

[5]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages
(1.5.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-
packages (from pandas) (2022.2.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
/opt/conda/lib/python3.10/site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.21.0 in /opt/conda/lib/python3.10/site-
packages (from pandas) (1.23.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[4]:
```
pip install statsmodels
```

```
Requirement already satisfied: statsmodels in /opt/conda/lib/python3.10/site-
packages (0.13.5)
Requirement already satisfied: packaging>=21.3 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (21.3)
Requirement already satisfied: scipy>=1.3 in /opt/conda/lib/python3.10/site-
packages (from statsmodels) (1.9.0)
Requirement already satisfied: pandas>=0.25 in /opt/conda/lib/python3.10/site-
packages (from statsmodels) (1.5.2)
Requirement already satisfied: patsy>=0.5.2 in /opt/conda/lib/python3.10/site-
packages (from statsmodels) (0.5.3)
Requirement already satisfied: numpy>=1.17 in /opt/conda/lib/python3.10/site-
packages (from statsmodels) (1.23.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging>=21.3->statsmodels)
(3.0.9)
Requirement already satisfied: python-dateutil>=2.8.1 in
/opt/conda/lib/python3.10/site-packages (from pandas>=0.25->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-
packages (from pandas>=0.25->statsmodels) (2022.2.1)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-packages
```

```
(from patsy>=0.5.2->statsmodels) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
pip install sklearn
```

```
Requirement already satisfied: sklearn in /opt/conda/lib/python3.10/site-
packages (0.0.post1)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.10/site-
packages (1.2.0)
Requirement already satisfied: joblib>=1.1.1 in /opt/conda/lib/python3.10/site-
packages (from scikit-learn) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/lib/python3.10/site-
packages (from scikit-learn) (1.23.2)
Requirement already satisfied: scipy>=1.3.2 in /opt/conda/lib/python3.10/site-
packages (from scikit-learn) (1.9.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (3.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[5]:
```python
# 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
```

1.1 Question 1

1(a)

[7]:
```python
bechdel = pd.read_csv('bechdel.csv')
bechdel.head()
```

[7]:
```
   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)

movie title

1(c)

[8]:
```
bechdel['title'].count()
```

[8]:
```
1794
```

1794 observations

1(d)

[8]:
```
pg = bechdel['rated'].value_counts()['PG']
pg_13 = bechdel['rated'].value_counts()['PG-13']
r = bechdel['rated'].value_counts()['R']
other = bechdel['rated'].value_counts()['other']
print('parental guidance suggested -', pg, 'parents strongly cautioned -',␣
↪pg_13, 'restricted -', r, 'other -', other)
```

```
parental guidance suggested - 257 parents strongly cautioned - 565 restricted -
691 other - 281
```

1.1.1 1(e)

In order for the sample to be representative of the population, it must be randomly selected and large enough. In this case, there is no information about whether the sample was randomly selected or not. In fact, it could have been quite otherwise, and many types of biases could have been involved, since the dataset uses only two websites as a source of information. Also, I wouldn't say it's big enough to represent the entire population of films. According to Google, there are about 500,000 films, which means that the sample is 0.3% (1794/500000) of the population.Therefore, I believe that the dataset is not representative of the movie set.

1(f)

[9]:
```
bechdel = bechdel.dropna(subset = ['runtime'])
bechdel.head()
```

[9]:
```
   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
```

1.2 Question 2

2(a)

[10]:
```python
mod = smf.ols(formula = 'imdb_rating ~ budget + drama + sci_fi + romcom', data␣
 ↪= bechdel).fit()
mod.summary()
```

[10]:
```
<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:                        10:39:16   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.
==============================================================================

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

2(b)

Adjusted R^2, 0.077.

2(c)

drama, romcom

2(d)

Holding all other variables constant, one unit increase in romcom is associated with -0.2111 units increase in the imdb ratings.

2(e)

100*0.0012 = 0.12 units increase in the imdb ratings

1.3 Question 3

3(a)

We've seen that at 5% level, budget and sci_fi failed to be significant, and we failed to conclude that there is a relationship between those features and IMDB. Also, since our R-squared value is low, which implies that our data is not fit well into our model. Therefore, I would say that the four features would be a good predictor of the IMDB.

3(b)

[11]:
```
x_df =  bechdel[['budget', 'drama', 'romcom', 'sci_fi']]
y_df = bechdel[['imdb_rating']]
x_df.head()
```

[11]:

	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

[13]:
```
x_df.head()
```

[13]:

	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
```

3(c)

[12]:
```
x_train, x_test, y_train, y_test = train_test_split(x_df,y_df,test_size = 0.2,␣
 ↪random_state=135)

x_train.head()
```

[12]:
```
          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
```

[15]:
```
len(x_train), len(x_test)
```

[15]:
```
(1272, 319)
```

3(d)

[13]:
```
regressor = LinearRegression()
regressor.fit(x_train, y_train)
print('Intercept', regressor.intercept_)        #we trained the algorithm! Now␣
 ↪print the intercept
print('Coefficient', regressor.coef_)
```

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

3(e)

intercept: 6.4645 and 6.42565479 budget: 0.0012 and 0.00161455 drama: 0.5445 and 0.58995507 sci-fi: -0.0378 and -0.23433976 romcom: -0.2111 and -0.0324223

the coefficient for drama in the sklearn regression model is higher than in q2.

3(f)

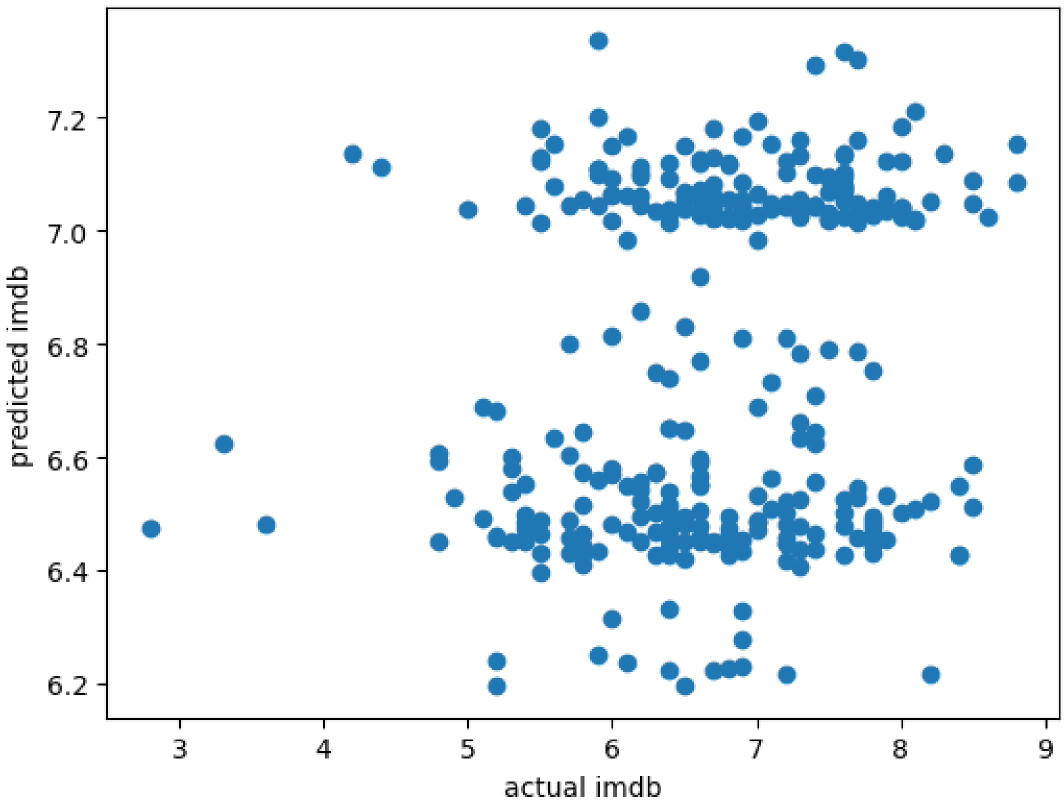
[14]:
```
y_pred = regressor.predict(x_test)
for i in range(10):
    print(y_pred[i])
```

```
[6.42969275]
[6.47394669]
[7.02198122]
[6.48700856]
```

```
[6.4566541]
[7.04467172]
[6.45022752]
[6.53049602]
[6.46014901]
[6.45018468]
```

3(g)

[15]:
```python
compare = pd.DataFrame({'Actual IMDB rating': y_test.to_numpy().flatten(),
                        'Predicted IMDB rating': y_pred.flatten()})
plt.scatter(y_test,y_pred)
plt.xlabel('actual imdb')
plt.ylabel('predicted imdb')
plt.show()
```



3(h)

[19]:
```python
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
 ↪y_pred)))
```

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

RMSE tells me the average difference between the predicted and actual values. In this case, on average, the actual and predicted values differ by 0.9284746856773564.

3(i)

[20]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('R-squared', metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 0.7351599189777014
Mean Squared Error: 0.8620652419436656
R-squared 0.0244054704545733
```

On average, IMDB shows that the IMDB score for a certain movie is 5, then the real IMDB score may be somewhere between ~4-6. I would not say that this is a very bad forecast, but also not the most accurate. In addition, we also have an R squared value of less than 3%, which makes me doubt the accuracy of the model. As a data scientist, I wouldn't use the model for important purposes.

1.4 Question 4

4(a)

[16]:
```
bechdel['bechdel'].describe()
```

[16]:
```
count     1591
unique       2
top       FAIL
freq       892
Name: bechdel, dtype: object
```

Fail: 892 observations

Pass: 1591-892=699 observations

4(b)

[22]:
```
(892/1591)*100
```

[22]:
```
56.06536769327467
```

56.06536769327467% of movies fail the test

4(c)

[17]:
```
x_df = bechdel[['year', 'budget', 'domgross', 'intgross',
'imdb_rating', 'romcom', 'drama', 'action', 'sci_fi']]
x_df
```

[17]:
```
      year     budget   domgross    intgross  imdb_rating  romcom  drama  \
2     2013  20.000000  53.107035  158.607035          8.3     0.0    1.0
```

```
3     2013   61.000000   75.612460  132.493015          6.8     0.0    0.0
4     2013   40.000000   95.020213   95.020213          7.6     0.0    1.0
5     2013  225.000000   38.362475  145.803842          6.6     0.0    0.0
6     2013   92.000000   67.349198  304.249198          5.4     0.0    0.0
…     …           …           …           …           …      …     …
1788  1971   14.386286   70.780525   70.780525          6.2     0.0    0.0
1789  1971  305.063707  404.702718  616.827003          6.6     0.0    0.0
1790  1971  143.862856   59.412143   64.760273          7.6     0.0    0.0
1791  1971   12.659931  236.848653  236.848653          7.8     0.0    0.0
1793  1970    5.997631   53.978683   53.978683          6.2     0.0    0.0

      action  sci_fi
2        0.0     0.0
3        1.0     0.0
4        0.0     0.0
5        1.0     0.0
6        1.0     0.0
…         …       …
1788     1.0     1.0
1789     1.0     0.0
1790     0.0     0.0
1791     1.0     0.0
1793     0.0     0.0

[1591 rows x 9 columns]
```

4(d)

[18]:
```
scaler = StandardScaler()
# x_df[['year','budget','domgross','intgross']] =
scaler.fit_transform(x_df[['year','budget','domgross','intgross']])
x_df.head()
```

[18]:
```
   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(e)

[19]:
```
y_df = bechdel[['bechdel']]
xtrain, xtest, ytrain, ytest = train_test_split(x_df,y_df,test_size = 0.2,␣
↪random_state=321)

xtrain.head()
```

[19]:
```
      year      budget    domgross     intgross  imdb_rating  romcom  drama  \
1203  2001  143.427209  415.208284  1167.441808          8.9     0.0    0.0
1599  1991   23.951765   59.415931    59.415931          6.6     0.0    1.0
585   2008   32.467689   68.368797   113.824188          7.3     1.0    1.0
1600  1990   71.319016  156.307443   434.511105          7.4     0.0    0.0
1551  1993  104.809829  135.525899   411.177020          6.3     0.0    0.0

      action  sci_fi
1203     1.0     0.0
1599     0.0     0.0
585      0.0     0.0
1600     0.0     1.0
1551     1.0     0.0
```

4(f)

[20]:
```
xtrain
```

[20]:
```
      year      budget    domgross     intgross  imdb_rating  romcom  drama  \
1203  2001  143.427209  415.208284  1167.441808          8.9     0.0    0.0
1599  1991   23.951765   59.415931    59.415931          6.6     0.0    1.0
585   2008   32.467689   68.368797   113.824188          7.3     1.0    1.0
1600  1990   71.319016  156.307443   434.511105          7.4     0.0    0.0
1551  1993  104.809829  135.525899   411.177020          6.3     0.0    0.0
…     …           …           …            …            …       …      …
1612  1990   53.489262  215.222722   357.486568          7.6     0.0    0.0
909   2005   34.001444    7.517458    18.435915          7.1     0.0    1.0
140   2012  147.122592  104.926572   311.393491          7.3     0.0    0.0
616   2008  248.918952  183.300049   640.362486          6.7     0.0    0.0
1141  2002   55.692706   31.641581    57.808305          5.0     1.0    0.0

      action  sci_fi
1203     1.0     0.0
1599     0.0     0.0
585      0.0     0.0
1600     0.0     1.0
1551     1.0     0.0
…        …       …
1612     1.0     0.0
909      0.0     1.0
140      0.0     0.0
```

```
616       1.0     0.0
1141      0.0     0.0

[1272 rows x 9 columns]
```

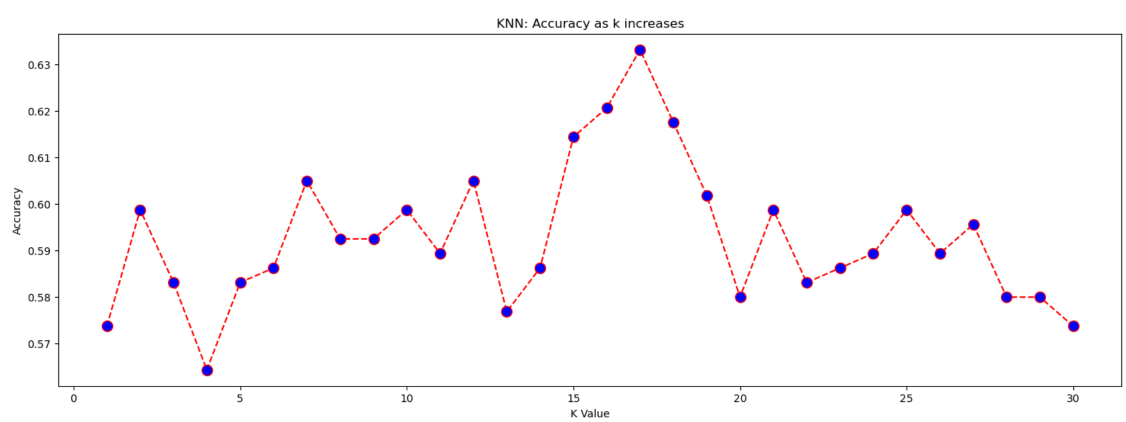
[21]:
```
error = []
accuracy = []

for i in range(1,31):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(xtrain, ytrain.to_numpy().flatten())
    pred_i = knn.predict(xtest)
    error.append(np.mean(pred_i != ytest.to_numpy()))
    accuracy.append(metrics.accuracy_score(ytest, pred_i))
```

[22]:
```
# plot accuracy

plt.figure(figsize=(18, 6))
plt.plot(range(1, len(accuracy)+1), accuracy, color='red', linestyle='dashed',␣
 ↪marker='o',
         markerfacecolor='blue', markersize=10)

# title and label axes
plt.title('KNN: Accuracy as k increases')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.show()
```



4(g)

From the graph, k = 17 have the greatest accuracy.

[23]:
```
classifier = KNeighborsClassifier(n_neighbors=17)                          #set␣
 ↪k=5 (a common value for k)
classifier.fit(xtrain, ytrain.to_numpy().flatten())                          ␣
 ↪ #train algorithm!  #was x_train, y_train
ypred = classifier.predict(xtest)
for i in range(6):
    print(ypred[i])
```

```
FAIL
PASS
PASS
PASS
FAIL
PASS
```

4(h)

[24]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

[26]:
```
print(confusion_matrix(ytest, ypred))
```

```
[[131  50]
 [ 67  71]]
```

50 movies were incorrectly predicted, although they have passed the Bechdel test.

4(i)

[27]:
```
print(classification_report(ytest, ypred))
```

```
              precision    recall  f1-score   support

        FAIL       0.66      0.72      0.69       181
        PASS       0.59      0.51      0.55       138

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

Recall for Pass is the portion of correct “positive” predictions for pass out of all positive predictions. It suggestts that we predicted ‘Fail’ a little better than ‘Pass’.

4(j)

With an accuracy of 63%, I would say that our model worked pretty well. Thus, on average, 6 out of 10 films would have been predicted correctly. Although I would prefer it to be slightly increased, but I think the results are useful.