Kyle Mahre

Final Project – Part 2

1. Objective:

I currently work in the entertainment industry; and in addition to that, I love the movies. As a movie fan. Sometimes, but not often, I find that I disagree with the general critical response or financial success of a film. For companies invested in content creation though; it is a hard truth that in order to consider a film a success, it must produce a profit that scales with its budget.

Box office blunders are a huge embarrassment to film makers. Flops can result from several unforeseeable causes, such as script changes, problems with actor contracts; or, they can result from general poor planning. The neural network I am going to build will take in data about movies and predict whether a film will be a flop or not.

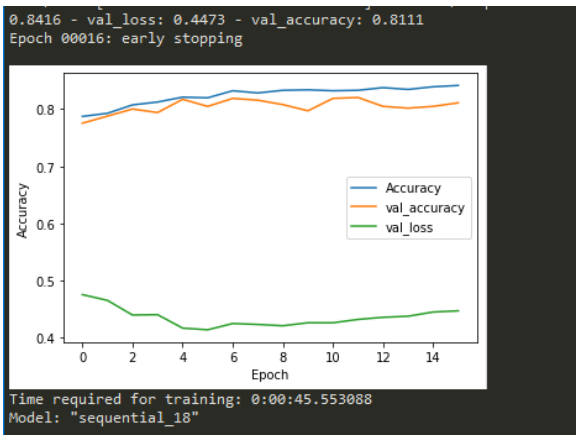
So far, in prepossessing the data, I have broken the films into budget categories and flagged the films according to their profit to budget ratio. Any film that made less in profit than it had as a budget is flagged as a flop. Other data I have selected to use as indicators have been release date, genres, runtime, popularity, and vote average and count. This data should help the neural net learn to predict according to seasonal patterns or change in consumer viewership tastes.

This sort of artificial neural net would be very useful to a content creator. I have modified the data preparation process so that the model can predict whether a film will belong to each of the success categories. After including the popularity and rating data and normalizing and transforming the data, the model successfully predicts if a movie will belong to a success class with 80 percent accuracy.

I went ahead and gathered ten more recent movies to attempt and predict which category they will likely fall. The model prediction of the recent films puts none of them into the dreaded flop class (loss). When I aimed the model to predict if a film is a super hit (profit greater than four times its budget), three films are flagged as likely superhits. It will take time to see how well the model works. It missed the mark on the recent bomb/flop of *Dark Phoenix*, predicting it would be a blockbuster. However; it successfully predicted *Joker* as a super-hit, and it has also flagged the upcoming animated film *Spies in Disguise* for great success.

Index	original_title
0	Little Women
1	1917
2	Spies in Disguise
3	Uncut Gems
4	Just Mercy
5	Joker
6	Frozen II
7	Dark Phoenix
8	Richard Jewell
9	Midway

	0
0	0
1	0
2	1
3	0
4	0
5	1
6	0
7	1
8	0
9	0



2. The Dataset:

I am using the Kaggle dataset located here: https://www.kaggle.com/tmdb/tmdb-movie-metadata. The genre and release date model I am improving upon can be found here: https://www.kaggle.com/diegoinacio/imdb-genre-based-analysis.

3. The code to the model will be submitted alongside this content.
4. My model is a feedforward ANN.
5. Once I improved the results of the neural net at predicting flops, I picked ten recent films to make predictions on. None of them were predicted to have very high chances of being a total flop. I pivoted to predicting the likelihood of other outcomes for each movie instead. The neural net doesn't always give the same answers each time I run it, but it does give fairly consistent results. For example, sometimes *Spies in Disguise* or *Just* Mercy are predicted to have a greater than 50 percent chance at becoming a super-hit while *Dark Phoenix* is not. Joker, however; always come up as a predicted super hit.

Index	original_title	0
0	Little Women	0.334695
1	1917	0.148387
2	Spies in Disguise	0.158338
3	Uncut Gems	0.124226
4	Just Mercy	0.366299
5	Joker	0.97859
6	Frozen II	0.00127545
7	Dark Phoenix	0.514694
8	Richard Jewell	0.0321904
9	Midway	0.00296152

0	Little Women	0.222624
1	1917	0.0777278
2	Spies in Disguise	0.297211
3	Uncut Gems	0.0672423
4	Just Mercy	0.482234
5	Joker	0.960433
6	Frozen II	0.0104556
7	Dark Phoenix	0.603681
8	Richard Jewell	0.0973759
9	Midway	0.000825277

Index	original_title	0
0	Little Women	0.248709
1	1917	0.102151
2	Spies in Disguise	0.00247771
3	Uncut Gems	0.0495696
4	Just Mercy	0.546478
5	Joker	0.968822
6	Frozen II	8.41022e-05
7	Dark Phoenix	0.298743
8	Richard Jewell	0.0731249
9	Midway	0.00053035

6. This whole model took at least ten hours (probably more) to build. Training the model doesn't take very long, usually taking only around a minute. I implemented an early stop when the model is training so that if after ten epochs the validation loss is still stuck, it stops the training process.

7.

Column Name	Data Type	Example	Use
low_budget	Binary Categorical	0 or 1	Predictor
mid_budget	Binary Categorical	0 or 1	Predictor
big_budget	Binary Categorical	0 or 1	Predictor
day	Numeric	18	Predictor
month	Numeric	12	Predictor
year	Numeric	2019	Predictor
GENRE_ACTION	Binary Categorical	0 or 1	Predictor
GENRE_ADVENTURE	Binary Categorical	0 or 1	Predictor
GENRE_ANIMATION	Binary Categorical	0 or 1	Predictor
GENRE_COMEDY	Binary Categorical	0 or 1	Predictor
GENRE_CRIME	Binary Categorical	0 or 1	Predictor
GENRE_DOCUMENTARY	Binary Categorical	0 or 1	Predictor
GENRE_DRAMA	Binary Categorical	0 or 1	Predictor
GENRE_FAMILY	Binary Categorical	0 or 1	Predictor
GENRE_FANTACY	Binary Categorical	0 or 1	Predictor
GENRE_FORIEGN	Binary Categorical	0 or 1	Predictor
GENRE_HISTORY	Binary Categorical	0 or 1	Predictor
GENRE_HORROR	Binary Categorical	0 or 1	Predictor
GENRE_MUSIC	Binary Categorical	0 or 1	Predictor
GENRE_MYSTERY	Binary Categorical	0 or 1	Predictor
GENRE_ROMANCE	Binary Categorical	0 or 1	Predictor
GENRE_SCIENCE_FICTION	Binary Categorical	0 or 1	Predictor
GENRE_THRILLER	Binary Categorical	0 or 1	Predictor
GENRE_WAR	Binary Categorical	0 or 1	Predictor
GENRE_WESTERN	Binary Categorical	0 or 1	Predictor
runtime	Numeric	130	Predictor
popularity	Numeric	15	Predictor
vote_average	Numeric	8.7	Predictor
vote_count	Numeric	5632	Predictor
flop	Percentage or Binary Categorical	.33 or 0	Response
bomb	Percentage or Binary Categorical	.61 or 1	Response
mediocre	Percentage or Binary Categorical	.05 or 0	Response
success	Percentage or Binary Categorical	.02 or 0	Response
blockbuster	Percentage or Binary Categorical	.51 or 1	Response
super_hit	Percentage or Binary Categorical	.95 or 1	Response

8. Data Preprocessing:

```
df = pd.read_csv("tmdb-movie-metadata/tmdb_5000_movies.csv", delimiter = ",", header = 0)

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Pretreat Data Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
def midbudge(budget):
    if (budget < 85000000 and budget >= 30000000):
        return True
    else:
        return False
df = df.drop_duplicates(['original_title'])
df = df[['id','original_title','budget', 'genres', 'release_date', 'revenue', 'runtime', 'popularity', 'vote_average','vote_count']].dropna()
df['budget'].dropna()
df['revenue'].dropna()
df.drop(df.loc[df['budget']==0].index, inplace=True)
df.drop(df.loc[df['revenue']==0].index, inplace=True)
df['profit'] = df['revenue']-df['budget']

df['low_budget'] = df['budget']<30000000
df['low_budget'] = df['low_budget'].astype(int)
df['mid_budget'] = df['budget'].apply(midbudge)
df['mid_budget'] = df['mid_budget'].astype(int)
df['big_budget'] = df['budget']>=85000000
df['big_budget'] = df['big_budget'].astype(int)
```

Above is the code for importing the data. First, duplicates are removed. Then I pull only the data I want, mostly numeric data other than the original titles and the genres. Then, any rows with either no data or a value of zero for budget and revenue data are dropped from the data-frame.

A new column, 'profit', equal to the result of subtracting budget from revenue, is created.

Finally, three new columns are created that are filled with binary values based on whether the data in this row corresponds to a film assigned to budget size categories.

```

df['flop'] = df['profit']<0
df['flop'] = df['flop'].astype(int)

df['bomb'] = df['profit']<.75*df['budget']
df['bomb'] = df['bomb'].astype(int)

df['mediocre'] = df['profit']>=.75*df['budget']
df['mediocre'] = df['mediocre'].astype(int)

df['success'] = df['profit']>1.5*df['budget']
df['success'] = df['success'].astype(int)

df['blockbuster'] = df['profit']>2.5*df['budget']
df['blockbuster'] = df['blockbuster'].astype(int)

df['super_hit'] = df['profit']>4*df['budget']
df['super_hit'] = df['super_hit'].astype(int)

```

Above, six new binary categorical columns are created for each row. If a film's profit meets a specified condition, it is assigned a value of 1, if not, it is assigned a value of 0.

```
df_genre = pd.DataFrame(columns = ['id','original_title','genre', 'cgenres', 'low_budget', 'mid_budget', 'big

def dataPrep(row):
    global df_genre
    d = {}
    genres = np.array([g['name'] for g in eval(row['genres'])])
    n = genres.size
    d['id'] = [row['id']]*n
    d['original_title'] = [row['original_title']]*n
    d['low_budget'] = [row['low_budget']]*n
    d['mid_budget'] = [row['mid_budget']]*n
    d['big_budget'] = [row['big_budget']]*n
    d['runtime'] = [row['runtime']]*n
    d['popularity'] = [row['popularity']]*n
    d['vote_average'] = [row['vote_average']]*n
    d['vote_count'] = [row['vote_count']]*n
    d['revenue'] = [row['revenue']]*n
    d['flop'] = [row['flop']]*n
    d['bomb'] = [row['bomb']]*n
    d['mediocre'] = [row['mediocre']]*n
    d['success'] = [row['success']]*n
    d['blockbuster'] = [row['blockbuster']]*n
    d['super_hit'] = [row['super_hit']]*n
    d.update(zip(('year', 'month', 'day'), map(int, row['release_date'].split('-'))))
    d['genre'], d['cgenres'] = [], []
    for genre in genres:
        d['genre'].append(genre)
        d['cgenres'].append(genres[genres != genre])
    df_genre = df_genre.append(pd.DataFrame(d), ignore_index=True, sort=True)

df.apply(dataPrep, axis=1)
df_genre = df_genre[['id','original_title','genre', 'low_budget', 'mid_budget', 'big_budget', 'revenue', 'day
df_genre = df_genre.infer_objects()
```

Above, this function makes a new row for all the genres (so they can all be accounted for, not just the first). Later, and before I run the model, I aggregate them all together grouped by original title name.

```
def unique(list1):
    x = np.array(list1)
    y = np.unique(x)
    z = y.tolist()
    print(z)
    return z

le = LabelEncoder()
onehot_encoder = OneHotEncoder(sparse=False)

genre_catagories = unique(df_genre['genre'])
integer_encoded = le.fit_transform(df_genre.loc[:,'genre'])
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
genrecats = pd.DataFrame(onehot_encoded, columns = [genre_catagories])
genrecats_sums = []

for i in range(0,len(genre_catagories)):
    genrecats_sums.append(sum(genrecats[(genre_catagories[i]),]))

for i in range(0,len(genrecats.columns)):
    newcolumn = genrecats[(genre_catagories[i],)]
    name_tail = genre_catagories[i]
    name_tail = name_tail.upper()
    name_tail = name_tail.split(' ')
    name_tail = '_'.join(name_tail)
    df['GENRE_'+name_tail] = newcolumn
    df_genre['GENRE_'+name_tail] = newcolumn

genrecats.to_csv('Genre_categories.csv')

df_genre = df_genre.groupby('original_title').agg({'id':'first','low_budget':'first', 'mid_budget': 'first',
                        'big_budget': 'first', 'day': 'first', 'month': 'first', 'year': 'first', 'GENRE_ACTION': 'sum',
                        'GENRE_ADVENTURE': 'sum', 'GENRE_ANIMATION': 'sum', 'GENRE_COMEDY': 'sum', 'GENRE_CRIME': 'sum',
                        'GENRE_DOCUMENTARY': 'sum', 'GENRE_DRAMA': 'sum', 'GENRE_FAMILY': 'sum', 'GENRE_FANTASY': 'sum',
                        'GENRE_FOREIGN': 'sum', 'GENRE_HISTORY': 'sum', 'GENRE_HORROR': 'sum', 'GENRE_MUSIC': 'sum',
                        'GENRE_MYSTERY': 'sum', 'GENRE_ROMANCE': 'sum', 'GENRE_SCIENCE_FICTION': 'sum',
                        'GENRE_THRILLER': 'sum', 'GENRE_WAR': 'sum', 'GENRE_WESTERN': 'sum', 'runtime': 'first',
                        'popularity': 'first', 'vote_average': 'first','vote_count': 'first','flop': 'first',
                        'bomb': 'first', 'mediocre': 'first', 'success': 'first', 'blockbuster': 'first', 'super_hit': 'first'}).reset_index()
```

```python
df_dataset = df_genre[['id','low_budget', 'mid_budget', 'big_budget', 'day', 'month',
                        'year', 'GENRE_ACTION', 'GENRE_ADVENTURE', 'GENRE_ANIMATION',
        'GENRE_COMEDY', 'GENRE_CRIME', 'GENRE_DOCUMENTARY', 'GENRE_DRAMA',
        'GENRE_FAMILY', 'GENRE_FANTASY', 'GENRE_FOREIGN', 'GENRE_HISTORY',
        'GENRE_HORROR', 'GENRE_MUSIC', 'GENRE_MYSTERY', 'GENRE_ROMANCE',
        'GENRE_SCIENCE_FICTION', 'GENRE_THRILLER', 'GENRE_WAR',
        'GENRE_WESTERN', 'runtime', 'popularity', 'vote_average', 'vote_count',
        'super_hit']]

dataset = df_dataset.values
#np.random.shuffle(dataset)
```

Depending on which of the dependent variables I want the neural net to predict, a different df_dataset can be instantiated. In the example above, the last variable is 'super_hit', and it will be used as the response/dependent variable.

I did this because it made the model more useful. Essentially, what the model does is predict the chance that a film given its popularity, budget, release date, and genres; should end up in each profit classification range. This system could be used by analysts to set budget limits and goals for films in each given class.

9. Metrics:

```python
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Define Model Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
start_time = datetime.datetime.now()
model = Sequential()
model.add(Dense(50, input_dim=30, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(50, activation='relu'))
model.add(Dense(1, activation = 'sigmoid'))

#compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Train Model Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
```

I didn't change my model's metrics from what I had in part one. The main improvement I made to improving the model's accuracy, was in adding and scaling more numeric predictor variables. Given the model's output, it appears that the neural net looks upon high vote counts more favorably than it does high average rating.

```python
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
estimator = model.fit(xtrain, ytrain, batch_size=1, epochs=100, verbose=1, validation_data=(xval,yval), callbacks=[es])
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Show output Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

plt.plot(estimator.history['accuracy'])
plt.plot(estimator.history['val_accuracy'])
plt.plot(estimator.history['val_loss'])
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Accuracy','val_accuracy','val_loss'], loc='best')
plt.show()
stop_time = datetime.datetime.now()
print ("Time required for training:",stop_time - start_time)
model.summary()
```

10. Validation:

```
233 X = dataset[:,0:30]
234 scaler = MinMaxScaler()
235 normalized = scaler.fit_transform(X)
236 inverse = scaler.inverse_transform(normalized)
237 Y = dataset[:,30]
238 xtrain = normalized[646:]
239 ytrain = Y[646:]
240 xval = normalized[:646]
241 yval = Y[:646]
242
```

Training and validation data split

After splitting data into training and validation sets and training the model, a confusion matrix and classification report are produced. This was helpful in verifying that the model was offering predictions and not simply going with the safest answer (at first the neural net would just say none of the movies were flops because they are generally pretty rare).

```
279
280 from sklearn.metrics import confusion_matrix
281 predictions = model.predict_on_batch(xval)
282 predictions[ predictions >= .5] = 1
283 predictions[ predictions < .5] = 0
284
285 print(confusion_matrix(yval, predictions))
286
287 from sklearn.metrics import classification_report
288 predictions = model.predict_on_batch(xval)
289 predictions[ predictions >= .5] = 1
290 predictions[ predictions < .5] = 0
291
292 print(classification_report(yval, predictions))
293
294 from keras.utils import plot_model
295
296 plot_model(model, show_shapes=True, to_file='model.png')
297
298 recent_films = pd.read_csv("recent_movies.csv", delimiter = ",", header = 0)
299
300 prediction_dataset = recent_films[['id','low_budget', 'mid_budget', 'big_budget', 'day', 'month',
301                     'year', 'GENRE_ACTION', 'GENRE_ADVENTURE', 'GENRE_ANIMATION',
302        'GENRE_COMEDY', 'GENRE_CRIME', 'GENRE_DOCUMENTARY', 'GENRE_DRAMA',
303        'GENRE_FAMILY', 'GENRE_FANTASY', 'GENRE_FOREIGN', 'GENRE_HISTORY',
304        'GENRE_HORROR', 'GENRE_MUSIC', 'GENRE_MYSTERY', 'GENRE_ROMANCE',
305        'GENRE_SCIENCE_FICTION', 'GENRE_THRILLER', 'GENRE_WAR',
306        'GENRE_WESTERN', 'runtime', 'popularity', 'vote_average', 'vote_count']]
307
308 normed_pred_dataset = scaler.fit_transform(prediction_dataset)
309 predict_recents = model.predict_on_batch(normed_pred_dataset)
310 #predict_recents[ predict_recents >= .5] = 1
311 #predict_recents[ predict_recents < .5] = 0
312
```

Recent film data

Calls on the model to predict on recent film data

```
[[288  62]
 [ 88 208]]
              precision    recall  f1-score   support

         0.0       0.77      0.82      0.79       350
         1.0       0.77      0.70      0.73       296

    accuracy                           0.77       646
   macro avg       0.77      0.76      0.76       646
weighted avg       0.77      0.77      0.77       646
```

True Positive / False Negative

False Positive / True Negative

One thing I noticed was that at the neural net was more accurate at predicting the chances a film will be a flop or a super-hit than it was the other classes.

11. Summary of output:

As shown in the opener, the when called to predict on data formatted like the model's it's output is the probability that the film will be in the class in question. For example, here is output for the recent film data chances of being a 'bomb':

Index	original_title		0
0	Little Women	0	0.121467
1	1917	1	0.0275865
2	Spies in Disguise	2	0.00159538
3	Uncut Gems	3	0.142656
4	Just Mercy	4	0.0538799
5	Joker	5	0.000224113
6	Frozen II	6	0.0802587
7	Dark Phoenix	7	1.49608e-05
8	Richard Jewell	8	0.393195
9	Midway	9	0.0674376

A film that is a 'bomb' is one that makes a profit less than 75% of its production budget. It still made money, but not enough to justify making it. Only one film of the ten recent films I chose has a decent chance of being a bomb according to the model on the above run.

12. In building this model I took what I understood to be standard practices as I understand them so far for predicting classes based on numeric input. The model is a feedforward neural network, using supervised learning (all the variables are known). I used a standard 80:20 training/validation data split. As there are 30 input dimensions, and the model is predicting whether that data will belong to a discrete class, there are 30 input units and only one output unit.
    I originally set the model to run at 100 epochs, but then set an early stop with a patience of 10 epochs, set to trigger if validation loss became stuck. The input data consists of 3228 rows of 30 variables. I didn't implement an adaptive learning rate or investigate momentum, as the model trained quick, the main issue was getting it to stop training once it stopped learning; which was addressed by early stopping.

13. Further Work to be done:

I would have liked to have found a way to get the model to learn to spot outliers like *Dark Phoenix*, which was supposed to be a super-hit as projected by industry analysts yet failed to achieve its promise. I think what I have ended up doing is discovering part of how industry analysts set projections that can by upset by reality. What this system does as of now is indicate whether a film punches above or below its budget and genre "weight class" when it comes to profit. For now, I am satisfied with that result, as before taking this course I had no knowledge on how to make a neural network that could do anything at all.

14. Code submitted with comments