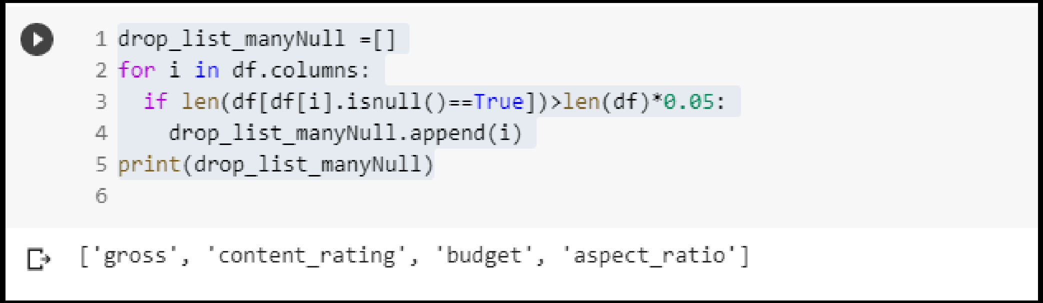
HW 10: ANN

By : 6338125721 Boonthicha Sae-jia

Code

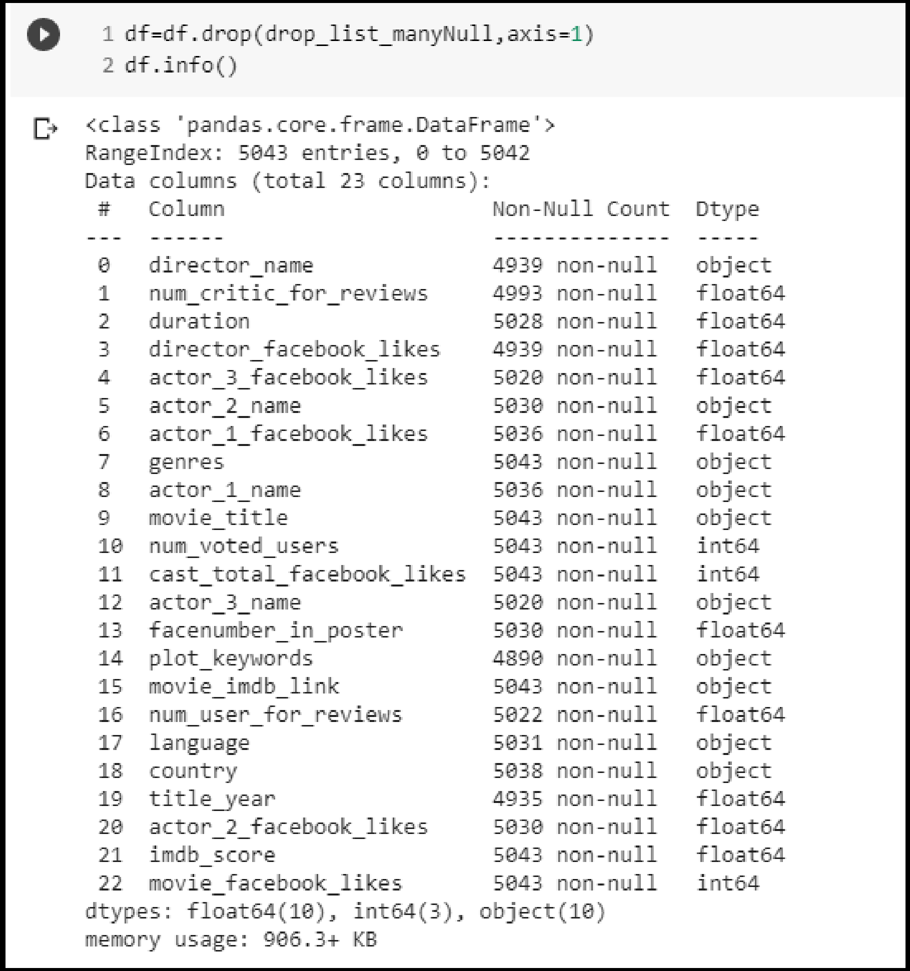
https://colab.research.google.com/drive/1FD9ZcCYiRkMcPAAi19Ij_HpcwN6tkAbM?usp=sharing

1 Data Preparation and Cleaning



```
drop_list_manyNull =[]
for i in df.columns:
  if len(df[df[i].isnull()==True])>len(df)*0.05:
    drop_list_manyNull.append(i)
print(drop_list_manyNull)

```

```
['gross', 'content_rating', 'budget', 'aspect_ratio']
```

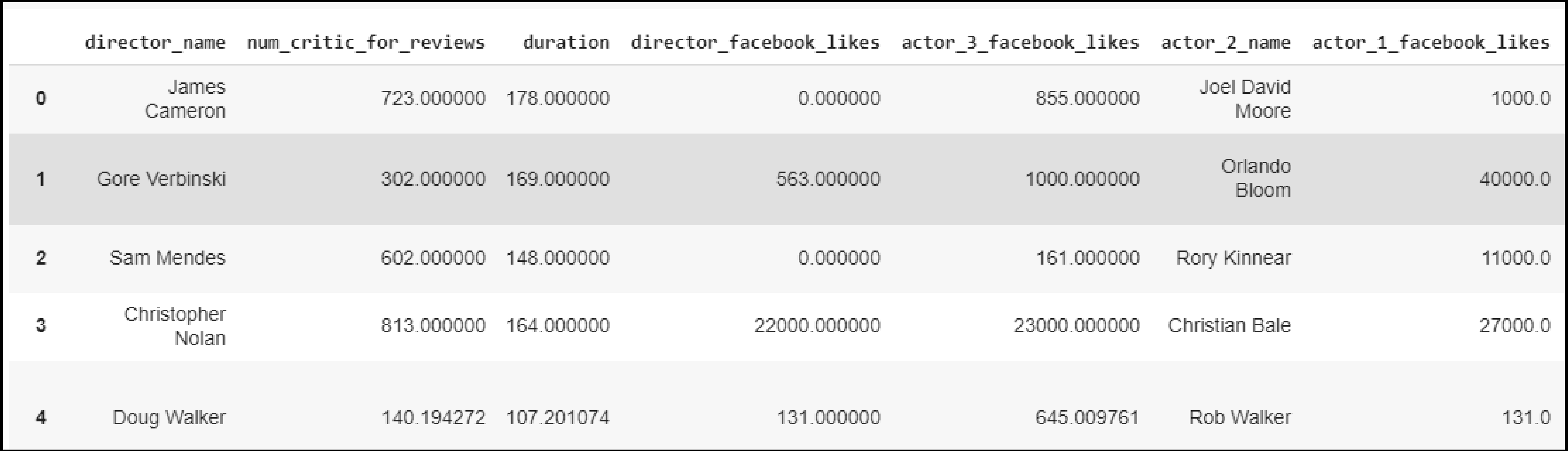


```
df=df.drop(drop_list_manyNull,axis=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   director_name              4939 non-null   object
 1   num_critic_for_reviews     4993 non-null   float64
 2   duration                   5028 non-null   float64
 3   director_facebook_likes    4939 non-null   float64
 4   actor_3_facebook_likes     5020 non-null   float64
 5   actor_2_name               5030 non-null   object
 6   actor_1_facebook_likes     5036 non-null   float64
 7   genres                     5043 non-null   object
 8   actor_1_name               5036 non-null   object
 9   movie_title                5043 non-null   object
 10  num_voted_users            5043 non-null   int64
 11  cast_total_facebook_likes  5043 non-null   int64
 12  actor_3_name               5020 non-null   object
 13  facenumber_in_poster       5030 non-null   float64
 14  plot_keywords              4890 non-null   object
 15  movie_imdb_link            5043 non-null   object
 16  num_user_for_reviews       5022 non-null   float64
 17  language                   5031 non-null   object
 18  country                    5038 non-null   object
 19  title_year                 4935 non-null   float64
 20  actor_2_facebook_likes     5030 non-null   float64
 21  imdb_score                 5043 non-null   float64
 22  movie_facebook_likes       5043 non-null   int64
dtypes: float64(10), int64(3), object(10)
memory usage: 906.3+ KB
```

a. Drop the columns that have missing value more than 5 percent

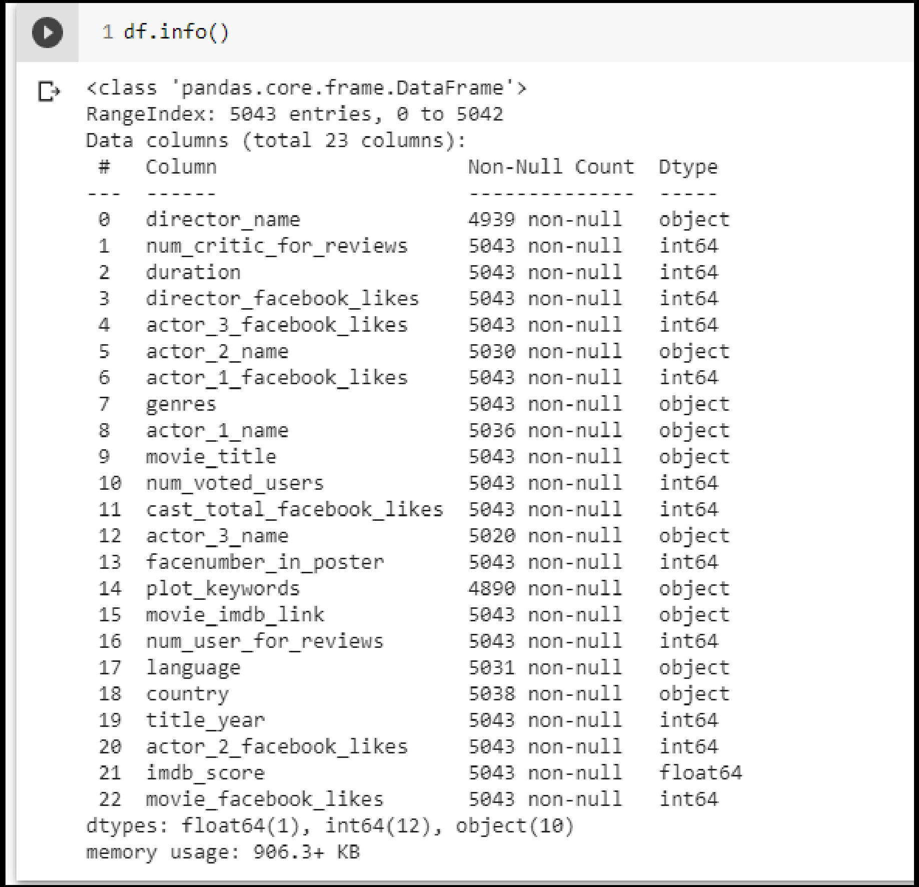
1 Data Preparation and Cleaning



	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes
0	James Cameron	723.000000	178.000000	0.000000	855.000000	Joel David Moore	1000.0
1	Gore Verbinski	302.000000	169.000000	563.000000	1000.000000	Orlando Bloom	40000.0
2	Sam Mendes	602.000000	148.000000	0.000000	161.000000	Rory Kinnear	11000.0
3	Christopher Nolan	813.000000	164.000000	22000.000000	23000.000000	Christian Bale	27000.0
4	Doug Walker	140.194272	107.201074	131.000000	645.009761	Rob Walker	131.0

b. Deal with missing values : use SimpleImputer() to change np.nan into mean of each column

1 Data Preparation and Cleaning

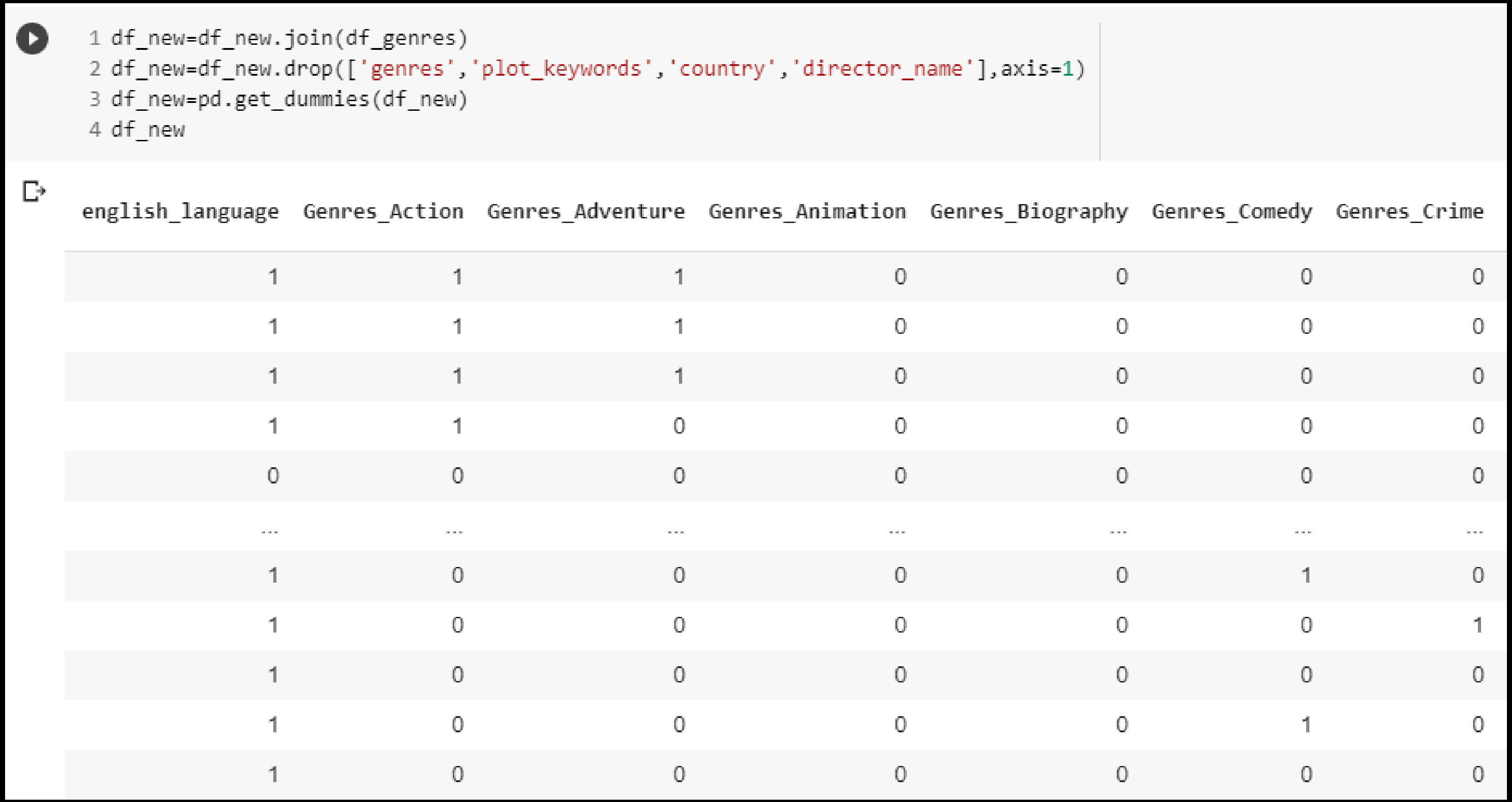


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   director_name              4939 non-null   object
 1   num_critic_for_reviews     5043 non-null   int64
 2   duration                   5043 non-null   int64
 3   director_facebook_likes    5043 non-null   int64
 4   actor_3_facebook_likes     5043 non-null   int64
 5   actor_2_name               5030 non-null   object
 6   actor_1_facebook_likes     5043 non-null   int64
 7   genres                     5043 non-null   object
 8   actor_1_name               5036 non-null   object
 9   movie_title                5043 non-null   object
 10  num_voted_users            5043 non-null   int64
 11  cast_total_facebook_likes  5043 non-null   int64
 12  actor_3_name               5020 non-null   object
 13  facenumber_in_poster       5043 non-null   int64
 14  plot_keywords              4890 non-null   object
 15  movie_imdb_link            5043 non-null   object
 16  num_user_for_reviews       5043 non-null   int64
 17  language                   5031 non-null   object
 18  country                    5038 non-null   object
 19  title_year                 5043 non-null   int64
 20  actor_2_facebook_likes     5043 non-null   int64
 21  imdb_score                 5043 non-null   float64
 22  movie_facebook_likes       5043 non-null   int64
dtypes: float64(1), int64(12), object(10)
memory usage: 906.3+ KB
```

c. Change continuous into discrete value in some column

1 Data Preparation and Cleaning

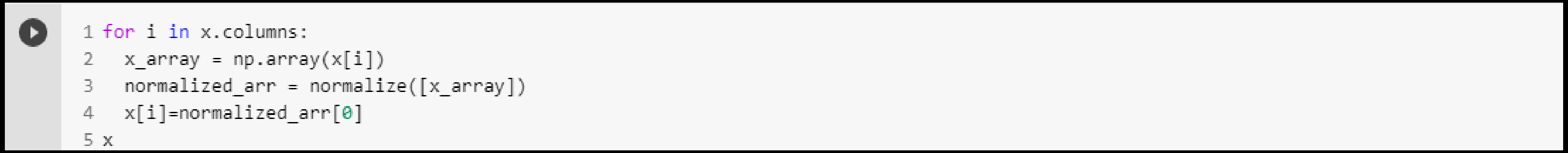


```
df_new=df_new.join(df_genres)
df_new=df_new.drop(['genres','plot_keywords','country','director_name'],axis=1)
df_new=pd.get_dummies(df_new)
df_new
```

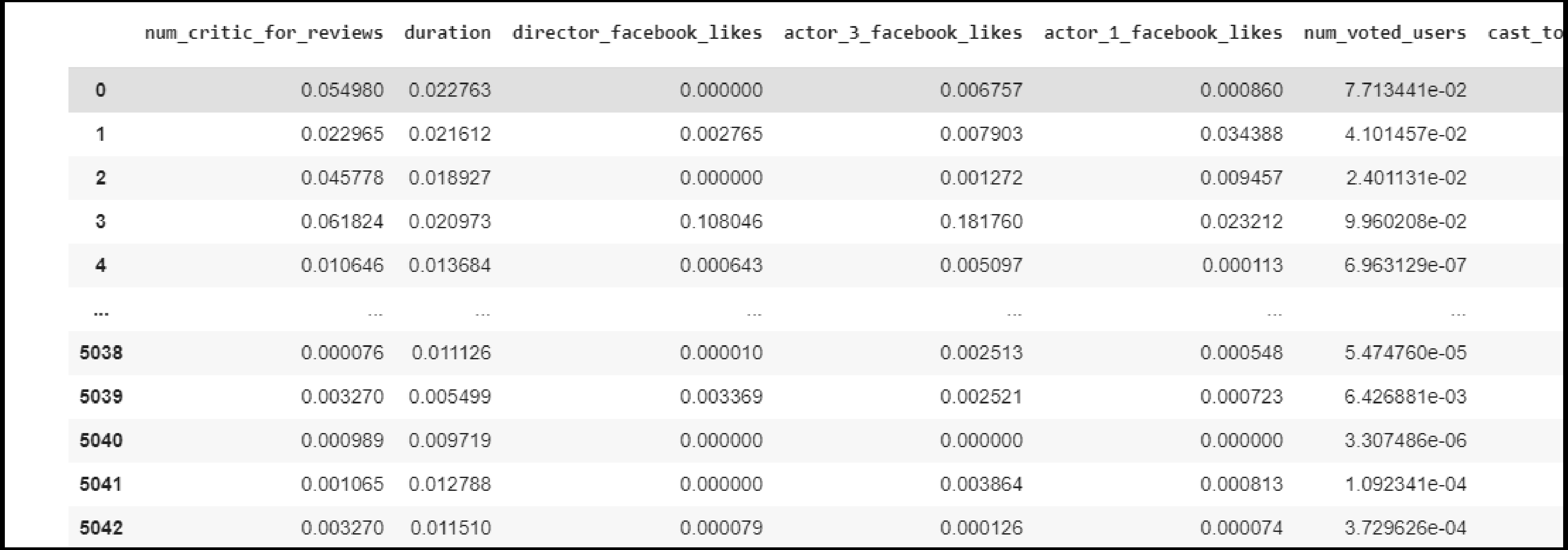
english_language	Genres_Action	Genres_Adventure	Genres_Animation	Genres_Biography	Genres_Comedy	Genres_Crime
1	1	1	0	0	0	0
1	1	1	0	0	0	0
1	1	1	0	0	0	0
1	1	0	0	0	0	0
0	0	0	0	0	0	0
...	...	...	...	...	...	...
1	0	0	0	0	1	0
1	0	0	0	0	0	1
1	0	0	0	0	0	0
1	0	0	0	0	1	0
1	0	0	0	0	0	0

d. Create some new features from get_dummies and rules

1 Data Preparation and Cleaning



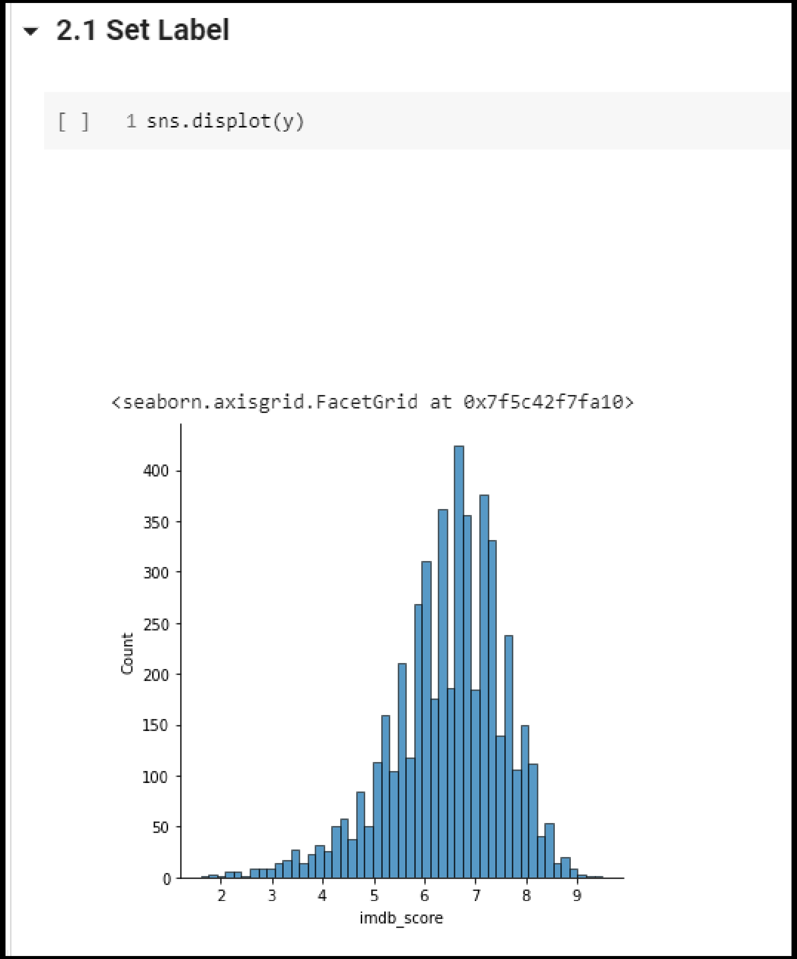
```python
for i in x.columns:
  x_array = np.array(x[i])
  normalized_arr = normalize([x_array])
  x[i]=normalized_arr[0]
x
```



	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	num_voted_users	cast_to
0	0.054980	0.022763	0.000000	0.006757	0.000860	7.713441e-02	
1	0.022965	0.021612	0.002765	0.007903	0.034388	4.101457e-02	
2	0.045778	0.018927	0.000000	0.001272	0.009457	2.401131e-02	
3	0.061824	0.020973	0.108046	0.181760	0.023212	9.960208e-02	
4	0.010646	0.013684	0.000643	0.005097	0.000113	6.963129e-07	
...	...	...	...	...	...	...	
5038	0.000076	0.011126	0.000010	0.002513	0.000548	5.474760e-05	
5039	0.003270	0.005499	0.003369	0.002521	0.000723	6.426881e-03	
5040	0.000989	0.009719	0.000000	0.000000	0.000000	3.307486e-06	
5041	0.001065	0.012788	0.000000	0.003864	0.000813	1.092341e-04	
5042	0.003270	0.011510	0.000079	0.000126	0.000074	3.729626e-04	

e. Normalized attributed

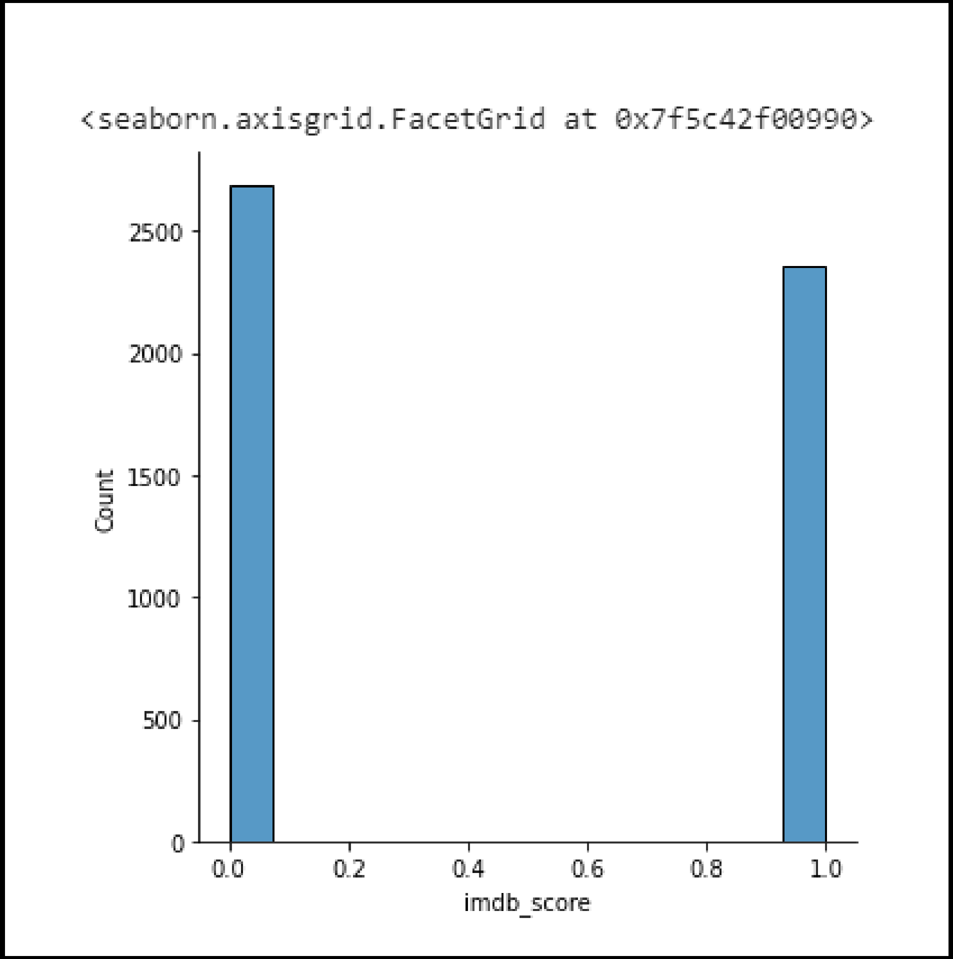
2 Data Visualization & Analysis



2.1 Set Label

```
[ ]  sns.displot(y)
```

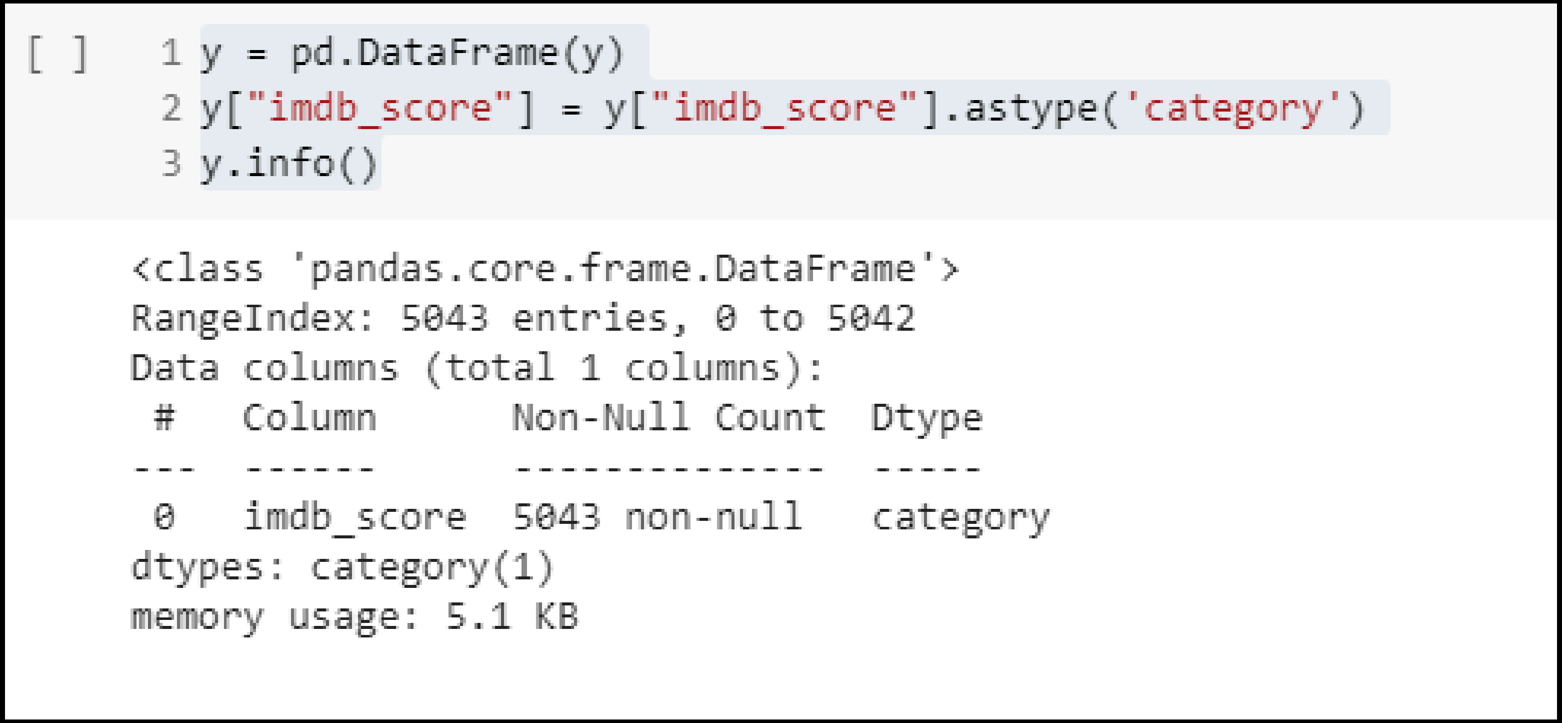
```
<seaborn.axisgrid.FacetGrid at 0x7f5c42f7fa10>
```



```
<seaborn.axisgrid.FacetGrid at 0x7f5c42f00990>
```

a. Create Distribution plot to see where we should divide the data

2 Data Visualization & Analysis

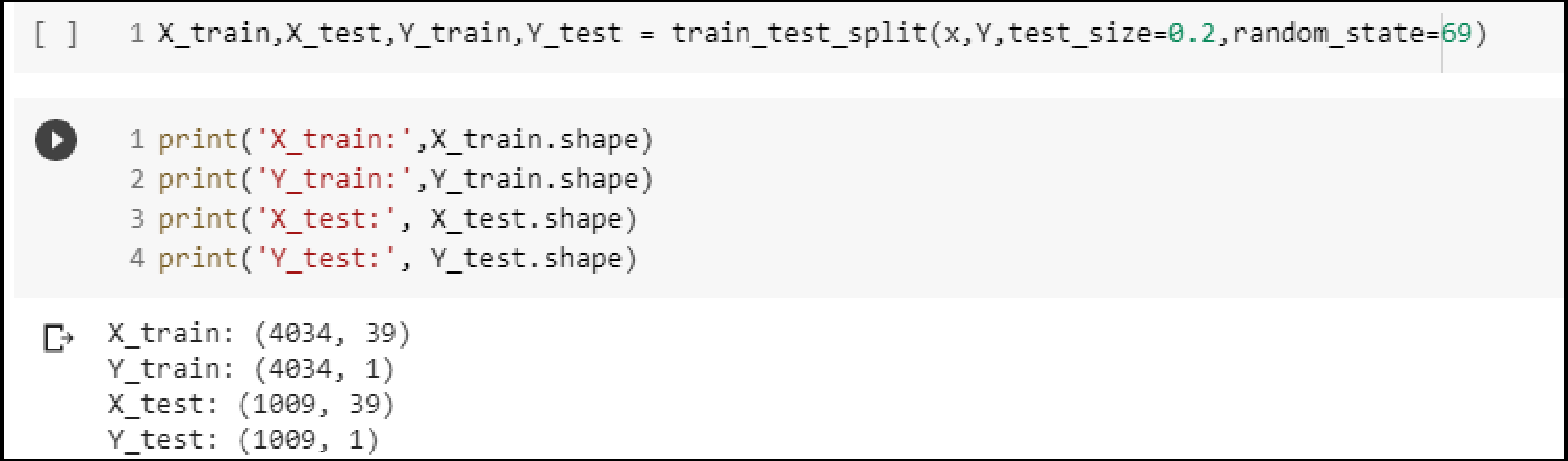


```
[ ]
y = pd.DataFrame(y)
y["imdb_score"] = y["imdb_score"].astype('category')
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   imdb_score  5043 non-null   category
dtypes: category(1)
memory usage: 5.1 KB
```

b. Change continuous value to category

3 Deep Learning Training



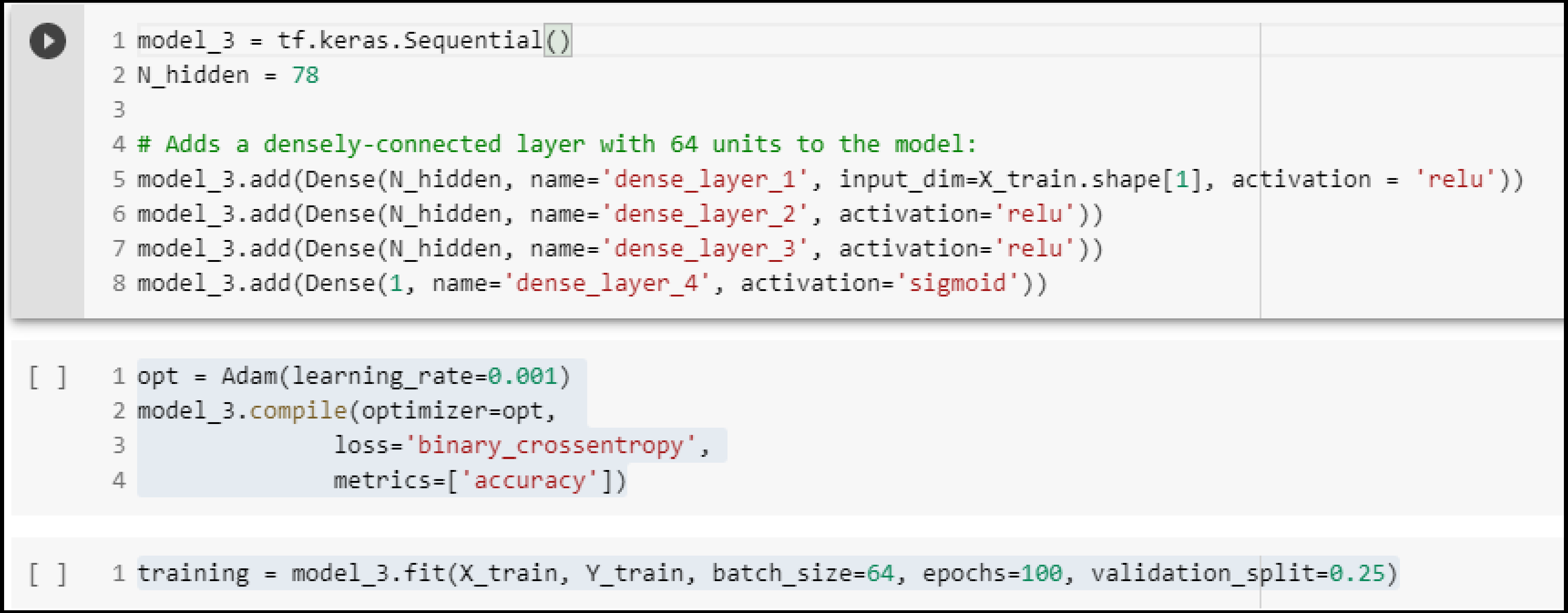
```
X_train,X_test,Y_train,Y_test = train_test_split(x,Y,test_size=0.2,random_state=69)
```

```
print('X_train:',X_train.shape)
print('Y_train:',Y_train.shape)
print('X_test:', X_test.shape)
print('Y_test:', Y_test.shape)
```

```
X_train: (4034, 39)
Y_train: (4034, 1)
X_test: (1009, 39)
Y_test: (1009, 1)
```

a. Split the dataset 80:20

3 Deep Learning Training



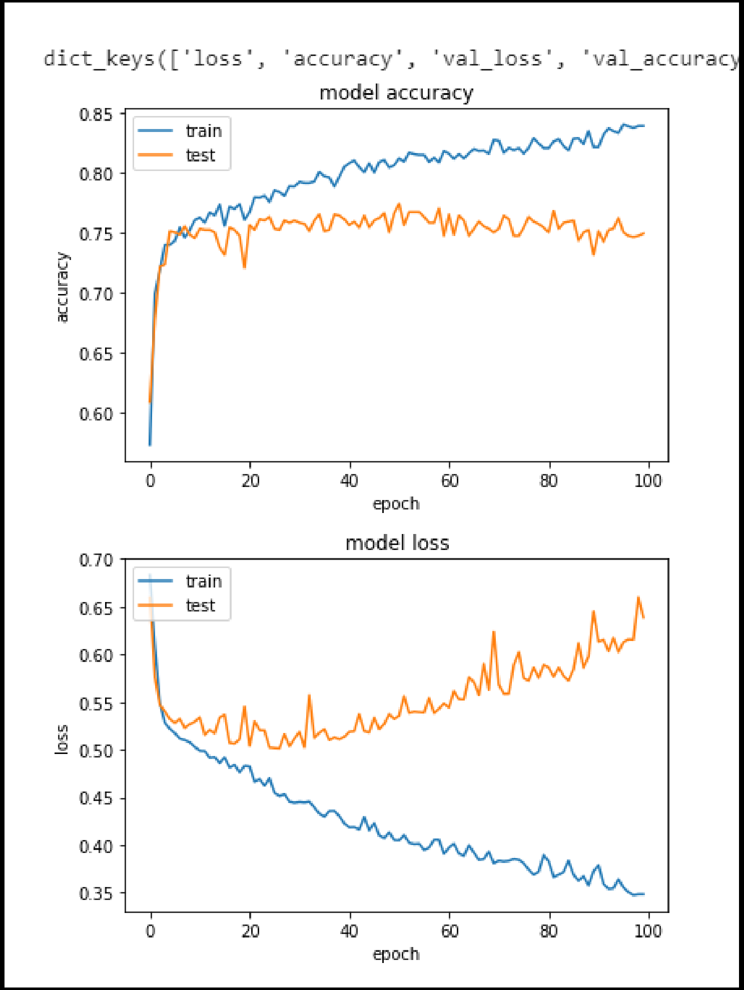
```
model_3 = tf.keras.Sequential()
N_hidden = 78

# Adds a densely-connected layer with 64 units to the model:
model_3.add(Dense(N_hidden, name='dense_layer_1', input_dim=X_train.shape[1], activation = 'relu'))
model_3.add(Dense(N_hidden, name='dense_layer_2', activation='relu'))
model_3.add(Dense(N_hidden, name='dense_layer_3', activation='relu'))
model_3.add(Dense(1, name='dense_layer_4', activation='sigmoid'))
```

```
[ ]
opt = Adam(learning_rate=0.001)
model_3.compile(optimizer=opt,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
[ ]
training = model_3.fit(X_train, Y_train, batch_size=64, epochs=100, validation_split=0.25)
```

b. Create and Training the models

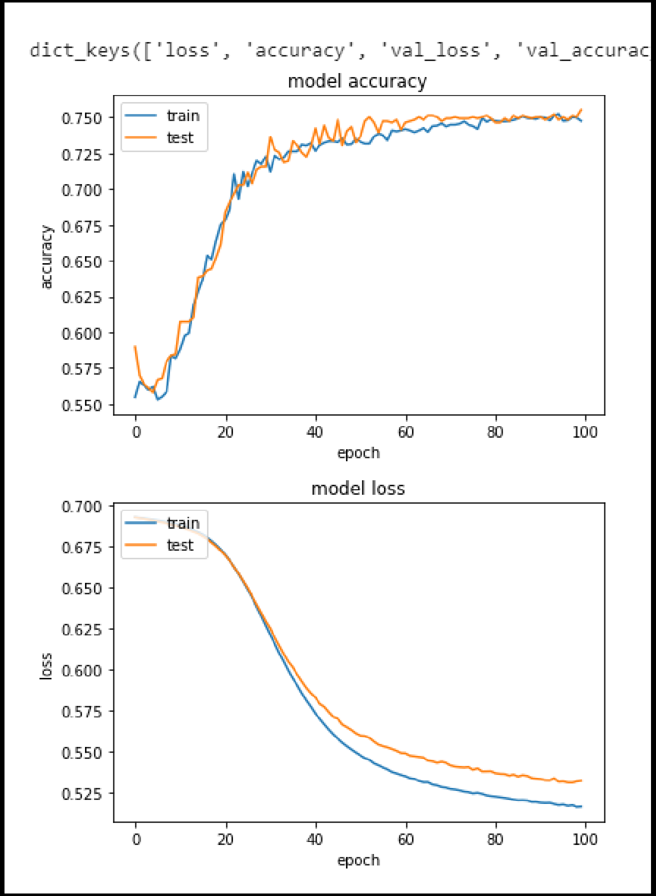
3 Deep Learning Training



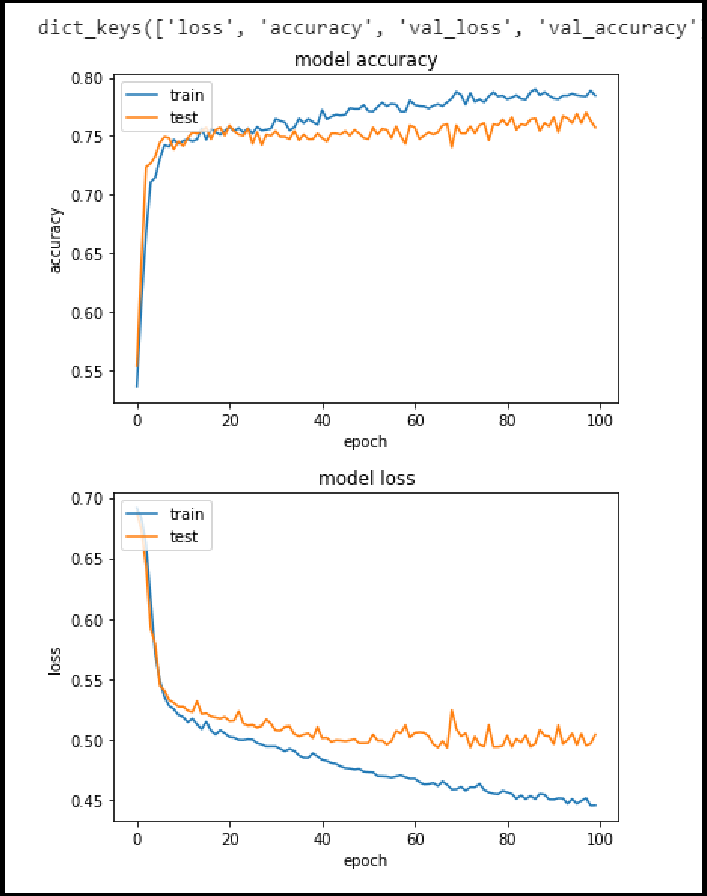
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy
```

b. Visualize to see the performance compare between train set and validation set

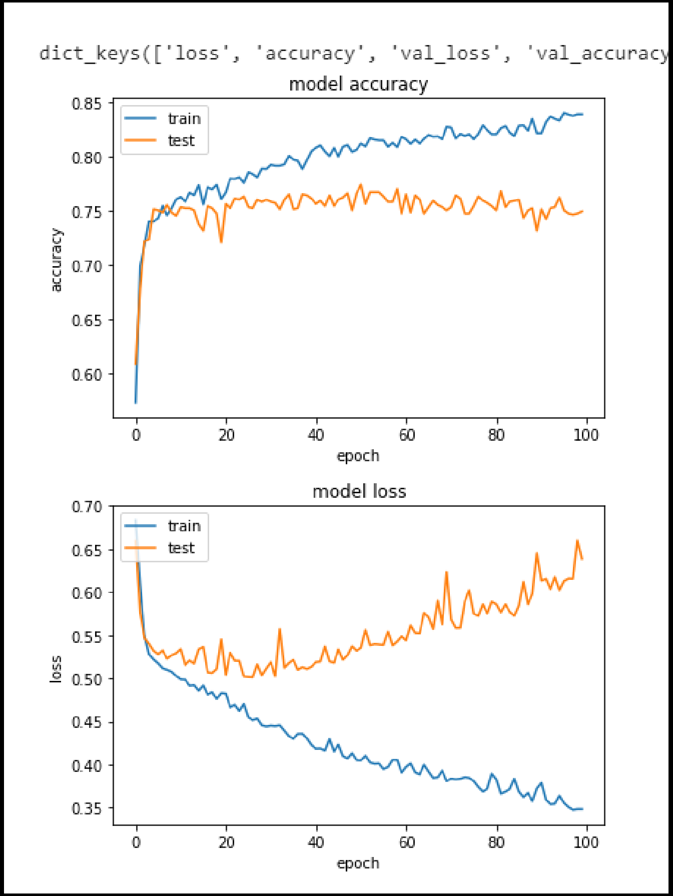
3 Deep Learning Training



```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accurac
```



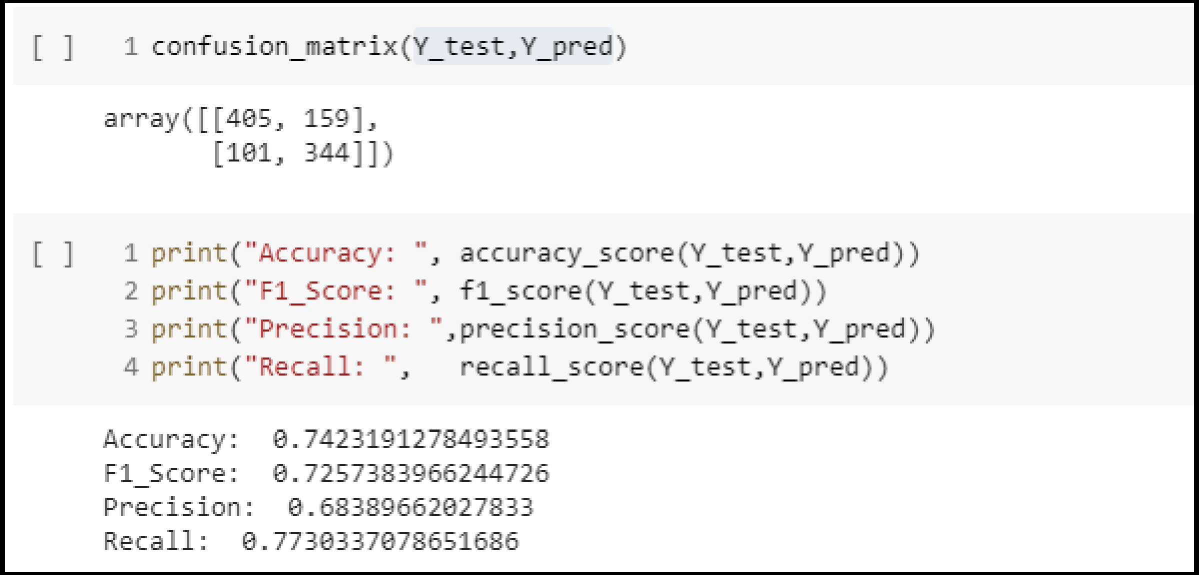
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'
```



```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy
```

c. Try some experiment to improve your accuracy of the model

4 Evaluation



```
[ ]  confusion_matrix(Y_test,Y_pred)

     array([[405, 159],
            [101, 344]])
```

```
[ ]  print("Accuracy: ", accuracy_score(Y_test,Y_pred))
     print("F1_Score: ", f1_score(Y_test,Y_pred))
     print("Precision: ",precision_score(Y_test,Y_pred))
     print("Recall: ",   recall_score(Y_test,Y_pred))

     Accuracy:  0.7423191278493558
     F1_Score:  0.7257383966244726
     Precision:  0.68389662027833
     Recall:  0.7730337078651686
```

a. After getting the best model from experiments, create confusion matrix to see the accuracy from test set