```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

Get Data

```
movie_df = pd.read_csv("industry_movies_5ratings_12months.csv", index_col=0)
numMonths = 12
movie_df["s"] = movie_df["s/c"]*np.sqrt(movie_df["d_height"]**2 + numMonths**2)
movie_df.columns.to_list()
```

```
feature_columns = [
 'Action', 'Adventure', 'Animation', 'Children',
 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Horror',
 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'Western',
 'budget',
 'rating',
 'votes',
 'runtime',
 'release',
 'profit',
 '0_pca',
 '1_pca',
 '2_pca',
 '3_pca',
 '4_pca',
 '5_pca',
 '6_pca',
 '7_pca',
 '8_pca',
 '9_pca',
 '10_pca',
 '11_pca',
 '12_pca',
 '13_pca',
 '14_pca',
 '15_pca',
 '16_pca',
```

```
'17_pca',
'18_pca',
'19_pca',
'20_pca',
'21_pca',
'22_pca',
'23_pca',
'24_pca',
'25_pca',
'26_pca',
'27_pca',
'28_pca',
'29_pca',
'30_pca',
'31_pca',
'32_pca',
'33_pca',
'34_pca',
'35_pca',
'36_pca',
'37_pca',
'38_pca',
'39_pca',
'40_pca',
'41_pca',
'42_pca',
'43_pca',
'44_pca',
'45_pca',
'46_pca',
'47_pca',
'48_pca',
'49_pca',
'50_pca',
'51_pca',
'52_pca',
'53_pca',
'54_pca',
'55_pca',
'56_pca',
'57_pca',
'58_pca',
'59_pca',
'60_pca',
'61_pca',
'62_pca',
'63_pca',
'64_pca',
```

```
 '65_pca',
 '66_pca',
 '67_pca',
 '68_pca',
 '69_pca',
 '70_pca',
 '71_pca',
 '72_pca',
 '73_pca',
 '74_pca',
 '75_pca',
 '76_pca',
 '77_pca',
 '78_pca',
 '79_pca',
 '80_pca',
 '81_pca',
 '82_pca',
 '83_pca',
 '84_pca',
 '85_pca',
 '86_pca',
 '87_pca',
 '88_pca',
 '89_pca',
 '90_pca',
 '91_pca',
 '92_pca',
 '93_pca',
 '94_pca',
 '95_pca',
 '96_pca',
 '97_pca',
 '98_pca',
 '99_pca',]

X = movie_df[feature_columns]
y_arc = movie_df[["s"]]
y_height = movie_df[["d_height"]]


X, y_arc, y_height = feature_scale(X, y_arc, y_height)


X_train_arc, X_test_arc, y_train_arc, y_test_arc = train_test_split(X, y_arc, test
X_train_height, X_test_height, y_train_height, y_test_height = train_test_split(X,
```

```
def feature_scale(X, y_arc, y_height):
  X_scaled = StandardScaler().fit_transform(X)
  y_arc_scaled = StandardScaler().fit_transform(y_arc)
  y_height = StandardScaler().fit_transform(y_height)
  return X_scaled, y_arc_scaled, y_height
```

Model Hyperparameters

```
epochs = 50
batch_size = 10
dropout = 0.3
validation_split = 0.1
alpha = 0.02
```

```
def make_model():
  model = keras.Sequential()
  model.add(layers.Dense(units=64, activation=tf.nn.relu, kernel_regularizer=tf.ke
  model.add(layers.Dropout(dropout))
  model.add(layers.Dense(units=64, activation=tf.nn.relu, kernel_regularizer=tf.ke
  model.add(layers.Dropout(dropout))
  model.add(layers.Dense(units=1)) # last layer no activation
  model.compile(optimizer=tf.optimizers.Adam(), loss='mse', metrics=["mse"])
  return model
```

```
def train_model(model, X_train, y_train):
  history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verb
  return history
```

```
def plotLoss(history):
  plt.plot(history.history["loss"])
  plt.plot(history.history["val_loss"])
  plt.title("Model Loss")
  plt.ylabel("Loss")
  plt.xlabel("Epoch")
  plt.legend(["Train", "Test"], loc="upper right")
  plt.show()
```

```
def plotAccuracy(history):
  plt.plot(history.history['accuracy'])
  plt.plot(history.history['val_accuracy'])
  plt.title('Model Accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Test'], loc='bottom right')
  plt.show()


arc_model = make_model()
height_model = make_model()
arc_history = train_model(arc_model, X_train_arc, y_train_arc)
height_history = train_model(height_model, X_train_height, y_train_height)


plotLoss(arc_history)
plotLoss(height_history)
#plotAccuracy(arc_history)
#plotAccuracy(height_history)


final_train_rmse_arc = np.sqrt(float(arc_history.history["mse"][-1]))
final_train_rmse_height = np.sqrt(float(height_history.history["mse"][-1]))

final_test_mse_arc = arc_model.evaluate(X_test_arc, y_test_arc)
final_test_rmse_arc = np.sqrt(final_test_mse_arc)
final_test_mse_height = height_model.evaluate(X_test_height, y_test_height)
final_test_rmse_height = np.sqrt(final_test_mse_height)

print("Final Train RMSE for Model One:", final_train_rmse_arc)
print("Final Train RMSE for Model Two:", final_train_rmse_height)
print("Final Test RMSE, accuracy for Model One:", final_test_rmse_arc)
print("Final Test RMSE, accuracy for Model Two:", final_test_rmse_height)
```