```
@@ -
1,11
+1,3
8 @@
                {
                "cells": [
                {
                "cell_type": "markdown",
                "metadata": {},
                "source": [
                "# Project: IMDB Movie Score Recommender"
                ]
                },
                {
                "cell_type": "markdown",
                "metadata": {},
                "source": [
                "# Question: How to recommend IMDB Movie Score(Score ranges from 0-Low to 10-Excellent)
                ]
                },
                {
                "cell_type": "markdown",
                "metadata": {},
                "source": [
                "## STEP-1 Purpose, motivation and description:\n",
                "\n",
                "1. What can we say about the success of a movie before it is released? Are there certa
                Given that major films costing over $100 million to produce can still flop, this questi
                "\n",
                "2. This question puzzled almost everybody for a long time since there is no universal
                critics to gauge the quality of a film, while others use their instincts. But it takes
                after a movie is released. And human instinct sometimes is unreliable.\n",
                "\n",
                "3. Predicting IMDB Score of a movie before it released in cinemas is my primary goal f
                human instincts data along with movie review sentiment analysis and character level lan
                "\n",
                "4. This will benefit all cinema lovers like me or film producers/directors who can get
                they are releasing like a pre-poll forecast."
                ]
                },
                {
                "cell_type": "code",
"execution_count": 1,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"#Importing all the libraries.\n",
@@ -24,12 +51,33 @@
"from sklearn.preprocessing import Imputer\n"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"## STEP-2 Data acquisition:\n",
"\n",
"Received the 50000 IMDB movie dataset from kaggle in a csv format. I was able to obtai
(998MB), spanning across 100 years in 66 countries. There are 2399 unique director name
"\n",
"## STEP-3 Data Cleaning:\n",
"\n",
"1. In almost all columns, I have missing values except the output variable “IMDB SCORE
good news.\n",
"\n",
"2. After careful analysis, I have found that I have only 2% of data loss if I remove 8
some fields, say actor facebook likes, director facebook likes and movie facebook likes
"\n",
"3. With pandas info () method, I have analyzed the data types for each features and co
save space in memory and faster processing. Such variables are imdb score (good or bad)
"\n",
"4. Used pandas describe () method to get descriptive statistics for all numerical feat
median, 25 and 755 percentiles. Luckily no outliers are found.\n",
"\n",
"5. Created a new output variable after discretizing the imdb_score to two categories B
"\n",
"6. Many machine learning algorithms require input features on the same scale for optim
budget before we can feed them to a model."
]
},
{
```

```
"cell_type": "code",
"execution_count": 2,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# Load the CSV file to a Pandas Dataframe\n",
@@ -41,7 +89,6 @@
"cell_type": "code",
"execution_count": 3,
"metadata": {
"collapsed": false,
"scrolled": true
},
"outputs": [
@@ -65,9 +112,7 @@
{
"cell_type": "code",
"execution_count": 4,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -98,7 +143,6 @@
"cell_type": "code",
"execution_count": 5,
"metadata": {
"collapsed": false,
"scrolled": true
},
"outputs": [
@@ -154,7 +198,6 @@
"cell_type": "code",
"execution_count": 6,
"metadata": {
```

```
"collapsed": false,
"scrolled": true
},
"outputs": [
@@ -189,9 +232,7 @@
{
"cell_type": "code",
"execution_count": 7,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -456,9 +497,7 @@
{
"cell_type": "code",
"execution_count": 8,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -504,12 +543,25 @@
"df.info()\n"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"## STEP-4 Exploratory data Analysis(EDA):\n",
"\n",
"After plotting histogram on the output variable imdb_score, i have found that distribu
scores are between 5.8(25%) and 7.2(75%) with std of 1.13.\n",
"\n",
"Plotted scatterplots between imdb_score and all other independent variables to find ou
correlated.\n",
"\n",
"Before we begin, it is necessary to investigate the correlation of those variables. Th
easily for my model.\n",
"\n",
"Refer to the Story_Telling_from_IMDB_Movie_dataset.ipynb for EDA's."
]
},
{
"cell_type": "code",
"execution_count": 9,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -556,9 +608,7 @@
{
"cell_type": "code",
"execution_count": 10,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -589,9 +639,7 @@
{
"cell_type": "code",
"execution_count": 11,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -623,9 +671,7 @@
{
```

```
"cell_type": "code",
"execution_count": 12,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -657,9 +703,7 @@
{
"cell_type": "code",
"execution_count": 13,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -691,9 +735,7 @@
{
"cell_type": "code",
"execution_count": 14,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stderr",
@@ -734,9 +776,7 @@
{
"cell_type": "code",
"execution_count": 15,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
```

```
"data": {
@@ -812,9 +852,7 @@
{
"cell_type": "code",
"execution_count": 16,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -886,9 +924,7 @@
{
"cell_type": "code",
"execution_count": 17,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -908,9 +944,7 @@
{
"cell_type": "code",
"execution_count": 18,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -930,9 +964,7 @@
{
"cell_type": "code",
"execution_count": 19,
"metadata": {
"collapsed": false
```

```
},
"metadata": {},
"outputs": [
{
"data": {
@@ -980,9 +1012,7 @@
{
"cell_type": "code",
"execution_count": 20,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# Define Function which will determine missing value percentages for any Data Fields\n
@@ -1005,9 +1035,7 @@
{
"cell_type": "code",
"execution_count": 21,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -1030,9 +1058,7 @@
{
"cell_type": "code",
"execution_count": 22,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -1078,9 +1104,7 @@
{
```

```
"cell_type": "code",
"execution_count": 23,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -1114,9 +1138,7 @@
{
"cell_type": "code",
"execution_count": 24,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -1146,9 +1168,7 @@
{
"cell_type": "code",
"execution_count": 25,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -1177,9 +1197,7 @@
},
{
"cell_type": "markdown",
"metadata": {
"collapsed": false
},
"metadata": {},
"source": [
"# check % of total # of rows we'll have to drop across all 4 cols"
```

```
]
@@ -1213,9 +1231,7 @@
{
"cell_type": "code",
"execution_count": 28,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"#Dataframe copied to a new one for dropping/manipulation purpose.\n",
@@ -1225,9 +1241,7 @@
{
"cell_type": "code",
"execution_count": 29,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# Writing a function which will determine % of rows dropped for these input and output
@@ -1280,9 +1294,7 @@
{
"cell_type": "code",
"execution_count": 30,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -1312,9 +1324,7 @@
{
"cell_type": "code",
"execution_count": 31,
"metadata": {
"collapsed": false
```

```
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -1349,9 +1359,7 @@
{
"cell_type": "code",
"execution_count": 32,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3325,9 +3333,7 @@
{
"cell_type": "code",
"execution_count": 33,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3357,9 +3363,7 @@
{
"cell_type": "code",
"execution_count": 34,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3390,7 +3394,6 @@
"cell_type": "code",
"execution_count": 35,
"metadata": {
"collapsed": false,
"scrolled": true
},
"outputs": [
@@ -3422,9 +3425,7 @@
{
"cell_type": "code",
"execution_count": 36,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3454,9 +3455,7 @@
{
"cell_type": "code",
"execution_count": 37,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3486,9 +3485,7 @@
{
"cell_type": "code",
"execution_count": 38,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3518,9 +3515,7 @@
```

```
{
"cell_type": "code",
"execution_count": 39,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3550,9 +3545,7 @@
{
"cell_type": "code",
"execution_count": 40,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3582,9 +3575,7 @@
{
"cell_type": "code",
"execution_count": 41,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3611,12 +3602,38 @@
"plt.scatter(df_copy['movie_facebook_likes'], df_copy['imdb_score'] )"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"## STEP-5 Feature Selection:\n",
"\n",
"1.Although initially I used 28 variables from IMDB website, many variables are not app
several critical variables.\n",
"\n",
"2.Correlation matrix shows that multicollinearity exists in the 15 continuous variable
multicollinearity.\n",
"\n",
"3.Therefore, I remove the following variables: \"gross\", \"cast_total_facebook_likes\
\"movie_facebook_likes\". \n",
"\n",
"4.Some variables are not applicable for prediction, such as \"num_voted_users\" and \"
unavailable before a movie is released.\n",
"\n",
"5.Used F-regression/chi2 for feature selection in machine learning pipeline from sklea
"\n",
"## STEP-6 Modeling:\n",
"\n",
"1.So far, we cleaned the data and did some exploratory analysis and did some testing t
"\n",
"2.Many machine learning algorithms require input features on the same scale for optima
budget before we can feed them to a model.\n",
"\n",
"3.Before we construct our first model pipeline, we divide the dataset into a separate
dataset (20 percent of the data).\n",
"\n",
"4.After discretizing the imdb_score to two categories Bad (0 to 7.5) and good (7.6 to
"and fit to different classification models say Decision Tree, Random Forest and Logist
accuracy with 18% error prediction rate for the test."
]
},
{
"cell_type": "code",
"execution_count": 42,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# Adding actor_1_facebook_likes, actor_2_facebook_likes & actor_3_facebook_likes toget
@@ -3628,9 +3645,7 @@
{
```

```
"cell_type": "code",
"execution_count": 43,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"df_copy['actor_facebook_likes_sum'] = actor"
@@ -3639,9 +3654,7 @@
{
"cell_type": "code",
"execution_count": 44,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"#Creating one new dataframe df_lm having only my input independent and dependent outpu
@@ -3652,9 +3665,7 @@
{
"cell_type": "code",
"execution_count": 45,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3762,9 +3773,7 @@
{
"cell_type": "code",
"execution_count": 47,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
```

```
"data": {
@@ -3789,9 +3798,7 @@
{
"cell_type": "code",
"execution_count": 48,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -3847,9 +3854,7 @@
{
"cell_type": "code",
"execution_count": 49,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3897,9 +3902,7 @@
{
"cell_type": "code",
"execution_count": 50,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -3920,9 +3923,7 @@
{
"cell_type": "code",
"execution_count": 51,
"metadata": {
"collapsed": false
```

```
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -3946,9 +3947,7 @@
{
"cell_type": "code",
"execution_count": 52,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -3978,9 +3977,7 @@
{
"cell_type": "code",
"execution_count": 53,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -4010,9 +4007,7 @@
{
"cell_type": "code",
"execution_count": 54,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -4042,9 +4037,7 @@
{
```

```
"cell_type": "code",
"execution_count": 55,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -4074,9 +4067,7 @@
{
"cell_type": "code",
"execution_count": 56,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -4105,9 +4096,7 @@
},
{
"cell_type": "markdown",
"metadata": {
"collapsed": false
},
"metadata": {},
"source": [
"Iteration 1 outcomes: My Linear Regression Iteration 1 gives me low score of 14% havin
actor_facebook_likes_sum(actor 1 + actor 2 + actor 3 facebook likes) and duration selec
4.57589045e-05 3.61228110e-06 1.51850886e-02]. I choose these variables as i see trends
"\n",
@@ -4118,19 +4107,15 @@
},
{
"cell_type": "markdown",
"metadata": {
"collapsed": false
},
"metadata": {},
"source": [
"# Iteration 2"
]
},
{
"cell_type": "code",
"execution_count": 57,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"df_iter2 = df.copy()"
@@ -4139,9 +4124,7 @@
{
"cell_type": "code",
"execution_count": 58,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# Taking care of missing data\n",
@@ -4161,9 +4144,7 @@
{
"cell_type": "code",
"execution_count": 59,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -4183,9 +4164,7 @@
{
"cell_type": "code",
"execution_count": 60,
"metadata": {
```

```
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# Encoding categorical data\n",
@@ -4216,9 +4195,7 @@
{
"cell_type": "code",
"execution_count": 61,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4263,9 +4240,7 @@
{
"cell_type": "code",
"execution_count": 62,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# split into a training and testing set. Selecting 30% for testing the Linear regressi
@@ -4277,9 +4252,7 @@
{
"cell_type": "code",
"execution_count": 63,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -4304,9 +4277,7 @@
```

```
{
"cell_type": "code",
"execution_count": 64,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4377,9 +4348,7 @@
{
"cell_type": "code",
"execution_count": 65,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -4456,9 +4425,7 @@
},
{
"cell_type": "markdown",
"metadata": {
"collapsed": false
},
"metadata": {},
"source": [
"# Multiple linear regression"
]
@@ -4494,9 +4461,7 @@
{
"cell_type": "code",
"execution_count": 67,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"df_mlr = df_mlr.fillna(0)"
@@ -4505,9 +4470,7 @@
{
"cell_type": "code",
"execution_count": 68,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"#score = pd.cut(df_mlr.imdb_score,2, labels=[0,1])\n",
@@ -4517,9 +4480,7 @@
{
"cell_type": "code",
"execution_count": 69,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"# Splitting the dataset into the Training set and Test set\n",
@@ -4533,9 +4494,7 @@
{
"cell_type": "code",
"execution_count": 70,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4565,9 +4524,7 @@
{
"cell_type": "code",
"execution_count": 71,
"metadata": {
```

```
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4585,9 +4542,7 @@
{
"cell_type": "code",
"execution_count": 72,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4664,9 +4619,7 @@
{
"cell_type": "code",
"execution_count": 73,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4701,9 +4654,7 @@
{
"cell_type": "code",
"execution_count": 74,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4721,9 +4672,7 @@
```

```
{
"cell_type": "code",
"execution_count": 75,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4773,9 +4722,7 @@
{
"cell_type": "code",
"execution_count": 76,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4807,9 +4754,7 @@
{
"cell_type": "code",
"execution_count": 77,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4830,6 +4775,25 @@
"print('Variance score: %.2f' % regressor.score(X_test, y_test))"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"## STEP-7 Evaluation:\n",
"\n",
"1. A model can suffer from under fitting (high bias) if the model is too simple or it
too complex for the underlying training data. Overfitting is a common problem in ML, wh
generalize well to unseen data (test data). Overfitting can be caused by too many param
hand, under fitting (high bias), which means that our model is not complex enough to ca
suffer from low performance on unseen data. To get a bias variance tradeoff, we need to
"\n",
"2. We used popular holdout method where we split the data into train, validation and t
validation is used for hyper parameter tuning or model selection. Separating out the te
generalize to new data. Disadvantages is that depending on the random data split, estim
"\n",
"3. Then I use K-fold cross validation with k=10 where we randomly split the training d
used for the model training and 1-fold is used for performance evaluation. This procedu
performance estimates. Average the performance.\n",
"\n",
"4. Since we are working with small training set, I increased the number of folds, that
result in low bias towards estimating the generalization performance by averaging the i
"\n",
"5. We evaluated my models using model accuracy, which is a useful metric with which to
precision, recall, F1 score evaluation using confusion matrix. The diagonal elements re
equal to the true label, while off-diagonal elements are those that are mislabeled by t
matrix the better, indicating many correct predictions.\n",
"\n",
"6. In machine learning, we have two type of parameters: those that are learned from th
and the hyper parameters that are optimized separately, for example regularization para
decision tree. I used grid search to help improve model performance by tuning hyperpara
learning rate C with value 0.01 can improve my accuracy score to 85% from 81% for both
]
},
{
"cell_type": "markdown",
"metadata": {},
@@ -4896,9 +4860,7 @@
{
"cell_type": "code",
"execution_count": 79,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stderr",
@@ -4926,9 +4888,7 @@
{
"cell_type": "code",
"execution_count": 80,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4960,9 +4920,7 @@
{
"cell_type": "code",
"execution_count": 81,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -4994,9 +4952,7 @@
{
"cell_type": "code",
"execution_count": 82,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5028,9 +4984,7 @@
{
"cell_type": "code",
"execution_count": 83,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5062,9 +5016,7 @@
{
"cell_type": "code",
"execution_count": 84,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5103,9 +5055,7 @@
{
"cell_type": "code",
"execution_count": 85,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -5197,9 +5147,7 @@
{
"cell_type": "code",
"execution_count": 86,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"score = pd.cut(df_mlr.imdb_score.sort_values(), 4, labels=['bad','medium','good','exce
@@ -5209,9 +5157,7 @@
{
"cell_type": "code",
"execution_count": 87,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"df_mlr['imdb_score'] = score"
@@ -5220,9 +5166,7 @@
{
"cell_type": "code",
"execution_count": 88,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -5383,9 +5327,7 @@
{
"cell_type": "code",
"execution_count": 90,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -5412,9 +5354,7 @@
{
"cell_type": "code",
"execution_count": 91,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5449,9 +5389,7 @@
{
"cell_type": "code",
"execution_count": 92,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5492,9 +5430,7 @@
{
"cell_type": "code",
"execution_count": 93,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5514,9 +5450,7 @@
{
"cell_type": "code",
"execution_count": 94,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5596,9 +5530,7 @@
{
"cell_type": "code",
"execution_count": 96,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5668,9 +5600,7 @@
{
"cell_type": "code",
"execution_count": 97,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -5701,9 +5631,7 @@
{
"cell_type": "code",
"execution_count": 98,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"def score_rating(a):\n",
@@ -5735,9 +5663,7 @@
{
"cell_type": "code",
"execution_count": 99,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5761,9 +5687,7 @@
},
{
```

```
"cell_type": "markdown",
"metadata": {
"collapsed": false
},
"metadata": {},
"source": [
"# Random Forest Classification"
]
@@ -5778,9 +5702,7 @@
{
"cell_type": "code",
"execution_count": 100,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5815,9 +5737,7 @@
{
"cell_type": "code",
"execution_count": 101,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5835,9 +5755,7 @@
{
"cell_type": "code",
"execution_count": 102,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -5856,9 +5774,7 @@
{
"cell_type": "code",
"execution_count": 103,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -5901,9 +5817,7 @@
{
"cell_type": "code",
"execution_count": 104,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"df_lgr= df[['imdb_score','director_facebook_likes','duration',\n",
@@ -5914,9 +5828,7 @@
{
"cell_type": "code",
"execution_count": 105,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -5938,9 +5850,7 @@
{
"cell_type": "code",
"execution_count": 106,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"#IMDB_score is divided into 0)bad movies) and 1(good movies) part.\n",
@@ -5951,9 +5861,7 @@
{
"cell_type": "code",
"execution_count": 107,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -5980,9 +5888,7 @@
{
"cell_type": "code",
"execution_count": 108,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"#Feature Scaling\n",
@@ -5995,9 +5901,7 @@
{
"cell_type": "code",
"execution_count": 109,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -6030,9 +5934,7 @@
{
"cell_type": "code",
"execution_count": 110,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -6050,9 +5952,7 @@
{
"cell_type": "code",
"execution_count": 111,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -6076,9 +5976,7 @@
{
"cell_type": "code",
"execution_count": 112,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -6104,9 +6002,7 @@
{
"cell_type": "code",
"execution_count": 113,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
```

```
@@ -6169,9 +6065,7 @@
{
"cell_type": "code",
"execution_count": 114,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [],
"source": [
"from sklearn.cross_validation import train_test_split\n",
@@ -6208,9 +6102,7 @@
{
"cell_type": "code",
"execution_count": 115,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"data": {
@@ -6344,9 +6236,7 @@
{
"cell_type": "code",
"execution_count": 116,
"metadata": {
"collapsed": false
},
"metadata": {},
"outputs": [
{
"name": "stdout",
@@ -6362,6 +6252,39 @@
"clf_l, Xtrain_l, ytrain_l, Xtest_l, ytest_l = do_classify(LogisticRegression(), {\"C\"
"df_lgr,
['director_facebook_likes','duration','actor_1_facebook_likes','actor_2_facebook_likes'
'budget'], 'imdb_score','Good')"
]
},
{
```

"cell_type": "markdown",
"metadata": {},
"source": [
"## STEP-8 Conclusion:\n",
"\n",
"1.Regression model can predict the actual imdb_score with less than 50% accuracy based
'director_facebook_likes','duration', 'actor_1_facebook_likes','actor_2_facebook_likes'
'facenumber_in_poster','title_year', 'budget'.\n",
"\n",
"2.Since the fitted Random Forest/Bayesian model explains more variability than that of
Random Forest/Bayesian to explain the insights found so far:\n",
"\n",
"The most important factor that affects movie rating is the duration. The longer the mo
"\n",
"Budget is important, although there is no strong correlation between budget and movie
"\n",
"The facebook popularity of director is an important factor to affect a movie rating.\n
"\n",
"The facebook popularity of the top 3 actors/actresses is important.\n",
"\n",
"The number of faces in movie poster has a non-neglectable effect to the movie rating.\
"\n",
"3.After discretizing the imdb_score to two categories Bad (0 to 5) and good(6 to 10) a
RandomForest, XGBoosting and Logistic Regression, i am getting 82% of average model pre
That's good actually.\n",
"\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"collapsed": true
},
"outputs": [],
"source": []
}
],
"metadata": {
@@ -6381,7 +6304,7 @@
"name": "python",
"nbconvert_exporter": "python",

```
"pygments_lexer": "ipython3",
"version": "3.6.0"
"version": "3.6.2"
}
},
"nbformat": 4,
```

68 changes: 66 additions & 2 deletions68 Sentiment Analysis of Movie Reviews using multilayer RNNs.ipynb

```
@@ -6,7 +6,9 @@
"source": [
"## Performing Sentiment analysis of movie reviews using multilayer RNNs.\n",
"\n",
"Sentiment analysis is concerned with analyzing the expressed opinion of a sentence or a
"Sentiment analysis is concerned with analyzing the expressed opinion of a sentence or a
"\n",
"# How to identify bad(0) or good
    (1)
    sentiment
    from Movie
    reviews?"
                          ]
                          },
                          {
                          @@ -16,6 +18,13 @@
                          "Movie dataset contains two columns, namely 'review' and 'sent
                          'sentiment' contains the '0' or '1' labels. The text component
                          we want to build an RNN model to process the words in each seq
                          1 classes."
                          ]
                          },
                          {
                          "cell_type": "markdown",
                          "metadata": {},
                          "source": [
                          "# Understanding RNN Modeling"
                          ]
                          },
                          {
```

```
"cell_type": "code",
"execution_count": 13,
@@ -254,6 +263,17 @@
"Image(filename='images/Unknown-13.png', width=600)"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"# STEP-1 Data acquisition:\n",
"\n",
"Received the 50000 IMDB movie dataset from kaggle in a csv fo
\"Sentiment\", where \"Review\" contains text of the movie rev
good or bad sentiment.\n",
"\n",
"The text component of the movie reviews are sequences of word
words in each sequence, and at the end classify the entire seq
]
},
{
"cell_type": "code",
"execution_count": 1,
@@ -291,6 +311,25 @@
"We create a mapping in the form of a dictionary that maps eac
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"## STEP-2 Data Cleaning:\n",
"\n",
"1. Movie reviews are sequence of words, therefore we want to
and at the end, classify the entire sequence to 0 or 1 classes
"\n",
"2. To prepare the data for input to neural network, we need t
the entire dataset.\n",
"\n",
"3. So far, we've converted sequence of words into sequence of
In order to generate input data that is compatible with our RN
```

```
sequences have the same length. Create matrix of zeroes where
index of words in each sequence from the right hand side of th
"\n",
"4. After we preprocess the dataset, we can proceed with split
"\n",
"5. Finally, we define one helper function that breaks a given
through these chunks(mini-batches).\n",
"\n",
"6. This is very useful techniques for handling memory limitat
dataset into mini-batches for training a neural network, rathe
them in memory during training."
]
},
{
"cell_type": "code",
"execution_count": 2,
@@ -432,7 +471,22 @@
"cell_type": "markdown",
"metadata": {},
"source": [
"# Building the RNN Model"
"# STEP-3 Modeling: Building the RNN Model"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"I have implemented SentimentRNN class that has the following
"\n",
"1. A constructor to set all the model parameters and then cre
to build the multilayer RNN.\n",
"\n",
"2. A build method that declares three placeholders for input
dropout configuration of the hidden layer. After declaring thi
RNN.\n",
"\n",
"3. A train method that creates a TensorFlow session for launc
batches of data, and runs for a fixed number of epochs, to min
also saves the model after 10 epochs for checkpointing.\n",
"\n",
"4. A predict method that creates a new session, restores the
carries out the predictions for the test data."
]
},
{
@@ -1205,6 +1259,16 @@
"proba = rnn.predict(X_test, return_proba=True)"
]
},
{
"cell_type": "markdown",
"metadata": {
"collapsed": true
},
"source": [
"# STEP-4 Conclusion:\n",
"RNN Sentiment model can predict the sentiment with 86% accura
]
},
{
"cell_type": "code",
"execution_count": null,
```