Movie Rating and Revenue Predictor

Bhagyesh Patil (16305R003)
Himanshu Singh (16305R005)
Satyaki Sen (16305R009)
Kapil Aggarwal (16305R010)

OBJECTIVE:

Objective of our project is to predict rating and revenue for a movie given its attributes like name of director, actors, budget, etc. This project can be an important utility for producers of movies while selecting directors and actors for their upcoming movies in order to maximize the profit. This project can also be used by common people to decide whether to watch the movie or not based on its predicted rating.

EXTRACTING DATASET:

The dataset is taken from Kaggle. Dataset consists of information of around 5000 movies available on IMDB. This dataset describes each movie by 28 attributes. Some of the attributes such as director name, actor name, genre etc. are in string format. But our mathematical model cannot accept string as a parameter. So, we will have to convert this data in a format that can be used in mathematical equations. Preprocessing phase describes how we changed string into numbers without altering the true meaning of the data.

PREPROCESSING:

1.) Some attributes do not impact the prediction of movie rating or revenue i.e. Name of the movie, IMDB link of the movie. So, we removed these attributes from the dataset.

2.) Some attributes are text i.e. name of the director, name of actors etc. So for this, we counted the total number of unique directors and then enumerated them from 1 to total count. Then, actual names of directors are replaced by their corresponding enumeration.

3.) There are total of 20 attributes to describe a movie in this dataset. These are: [Action, Adventure, Fantasy, Sci-Fi, Thriller, Romance, Animation, Comedy, Family, Fantasy, Musical, Mystery, Western, Drama, Sport, Crime, Horror, History, War, and Biography]. A movie can fall into more than one genre. Hence, we used one hot encoding to convert this into number. The current column that consisted of more than one genre in this form genre**i** | genre**j** | genre**k** | genre**l** is replaced with 20 columns (one column for each genre). These columns have 1 if movie falls into this genre else 0.
In this case, ith, jth, kth and lth columns will have 1 and rest will have 0 for this data point.

4.) Some attribute values are empty in the dataset for some movie. These missing fields are filled by the average of filled entries in corresponding column.

APPROACHES:

Dataset is divided into train and test set in the ratio of 4:1. Now, we can think of two approaches. In the first approach, dependent variable is assumed to be a continuous variable and in the other approach dependent variable is assumed as a discrete variable. When movie rating is to be predicted then independent variables are all attributes in the dataset except rating and dependent variable is rating. And when movie revenue is being predicted then dependent variables are all variables except revenue and dependent variable is revenue.

1. Regression

We assumed dependent variable i.e. movie rating and movie revenue as continuous variable. Following algorithms have been used: 1) Linear Regression, 2) Support Vector Regression – using Radial Basis Function, 3) Kernelized Ridge Regression and 4) Lasso Regression

2. Classification

We assumed dependent variable as discrete variable. Considering movie rating can take a value from the set {1.0, 1.1...1.9, 2.0, 2.1,...,8.9, 9.1, 9.2...10.0} i.e. a total of 101 class labels. For movie revenue, we divided revenue into ranges of [1-2], [2-3], [3-4]... [99,100] millions of dollars. While predicting revenue, algorithms will try to make prediction such that revenue falls in any one of these classes. Following classification algorithms are implemented: 1) Linear classifier 2) Support Vector Machine – using Radial Basis Function and 3) Deep Neural Network.

ALGORITHMS:

1. Regression:

1a. Linear Regression:

Our first approach is to apply linear regression. Given a data set D(x(i),y(i)) of size n, a linear regression model states that the relationship between the dependent variable *y(i)* and the regressors *x(i)* is linear. This relationship is modeled as
y(i) = x(i)(0)*w(0) + x(i)(1)*w(1) … x(i)(p)*w(p) + b , where w is a vector of size p and b is a bias. This is linear regression because it is linear in terms of weight vector. We have

used inbuilt library for this.

```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
y_predicted = regr.fit(X_train, Y_train).predict(X_test)
```

Then we calculated mean squared error which is 0.57 (approx) for rating prediction by using numpy.mean((y_predicted-Y_test)**2) function and 2.64498074974e+15 for revenue prediction. For rating it performs well but for revenue it has not given good results.

1b. SVR RBF (Support Vector Regression - Radial Basis Function):

SVR tries to minimize the following error:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\vec{w}\cdot\vec{x}_i - b)\right)\right] + \lambda\|\vec{w}\|^2,$$

We implemented this by using the following library:

```
svr_rbf= SVR(kernel='rbf', C=0.1, gamma=0.2)
y_predicted = svr_rbf.fit(X_train, Y_train).predict(X_test)
```

And mean square error is calculated using following function:

```
print(np.mean((y_predicted-Y_test_rev)**2)).
```

In this case, it comes out to be 0.84 (approx) for rating and 6.74739639091e+15 for revenue prediction which is not good.

1c. Kernelized Ridge Regression:

In ridge regression the problem is to minimize the following error:

$$\sum_i \left(\mathbf{x}_i \cdot \mathbf{w} - y_i\right)^2 + \lambda \mathbf{w}\cdot\mathbf{w}$$

After applying KKT conditions, decision function boils down to

$$f(\mathbf{x}) = \mathbf{x}\cdot\mathbf{w} = -\frac{1}{\lambda}\sum_i \alpha_i \mathbf{x}\cdot\mathbf{x}_i.$$

Which can be represented in kernelized form

$$f(\mathbf{x}) = -\frac{1}{\lambda}\sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i).$$

where alpha is Lagrange multiplier. We used inbuilt library to implement this algorithm in our project as follows:
krr = KernelRidge(alpha=0.1) #Create an object of Kernel Ridge Regression
y_krr = krr.fit(X_train_rev, Y_train_rev).predict(X_test_rev) #Train and predict
print(np.mean(abs(y_krr-Y_test_rev)**2)) # Calculate mean square error
MSE for rating prediction is 0.77 (approx) and 35705584.9045 for revenue. For revenue prediction, kernelized ridge regression is performing better.

1d. Lasso Regression:

Constrained objective to minimize is :

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\} \text{ subject to } \|\beta\|_1 \leq t.$$

Which boils down to following dual objective (Penalized form):

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

Where the exact relationship between t and lambda is data dependent.
We have implemented this algorithm using inbuilt library as follows:
lasso = linear_model.Lasso(lambda=0.1)
y_lasso = lasso.fit(X_train_rate, Y_train_rate).predict(X_test_rate)
print(np.mean(abs(y_lasso-Y_test_rate)))
MSE for revenue is 33809939.8432 and for rating it is 0.71.

2. Classification:

2a. SVM RBF (Support Vector Machine - Radial Basis Function):

Implementation using inbuilt function is as follows:
clf = svm.SVC(kernel='rbf', gamma=0.7, C=1.0).fit(X_train_rate, Y_train_rate.astype('int'))
y_predicted = clf.predict(X_test_rate)
The *C* parameter is a regularization/slack parameter and gamma is a measure of spread.

```
print(metrics.classification_report(Y_test_rate.astype('int'), y_predicted))

             precision    recall  f1-score   support

          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         8
          3       0.00      0.00      0.00        24
          4       1.00      0.04      0.08        75
          5       1.00      0.00      0.01       242
          6       0.36      1.00      0.53       439
          7       1.00      0.02      0.04       377
          8       1.00      0.03      0.05        77
          9       0.00      0.00      0.00         1

avg / total       0.75      0.36      0.21      1245

print("Confusion matrix")
print(metrics.confusion_matrix(Y_test_rate.astype('int'), y_predicted))

[[  0   0   0   0   0   2   0   0   0]
 [  0   0   0   0   0   8   0   0   0]
 [  0   0   0   0   0  24   0   0   0]
 [  0   0   0   3   0  72   0   0   0]
 [  0   0   0   0   1 241   0   0   0]
 [  0   0   0   0   0 439   0   0   0]
 [  0   0   0   0   0 369   8   0   0]
 [  0   0   0   0   0  75   0   2   0]
 [  0   0   0   0   0   1   0   0   0]]
```

2b. Deep Neural Network:

Will be implemented in later.

FUTURE WORK:

We will be implementing cross-validation to predict better, grid search to find C and gamma in SVM.

CONCLUSION:

As of now, linear regression seems to perform better for rating prediction and Lasso regression is better for revenue prediction. But this is certainly not the final result because lots of tuning of parameter needs to be done while cross validation and grid search.