Team Performance Predictor

Aditya Prasad Mishra
CIDSE, Arizona State University
ASU ID: 1211165121
User ID:amishr28
amishr28@asu.edu

Shalmali Bhoir
CIDSE, Arizona State University
ASU ID: 1209395067
User ID: sbhoir
sbhoir@asu.edu

Manikandan Anandan
CIDSE, Arizona State University
ASU ID: 1210414345
User ID: manandan
manandan@asu.edu

Piyush Nolastname
CIDSE, Arizona State University
ASU ID: 1211198375
User ID: ppapreja
ppapreja@asu.edu

Pragna Munagala
CIDSE, Arizona State University
ASU ID: 1211111639
User ID: pmunagal
pmunagal@asu.edu

Rohit Balachandran Menon
CIDSE, Arizona State University
ASU ID: 1212386823
User ID: rmenon9
rmenon9@asu.edu

April 29, 2017

Abstract

Forming a high-performance team has always been a fundamental challenge. The difficulty often lies in creating such teams from a network of individuals. The idea here is to consider individual skills and past collaboration efforts to predict the outcome of any future collaboration. In this project, we aim to utilize the Movielens dataset to analyze the network of individuals which consists of actors and directors. The main factors considered here are each individual's overall skill, skills in different movie genres and collaboration amongst each pair of individuals i.e. actors and directors. based on their skills in different movie genres and overall collaboration amongst them to predict the performance of the team in terms of Movie Rating. We implement and study different classification models such as Single Layer Neural Network, Multi-Layer Neural Network and Gaussian Naive Bayes model. We also explore effect of different parameters involved in theses algorithms on the accuracy of the results. We then compare the results of these developed models to other classification models like SVM, Random Forests, Logistic Regression etc. using Scikit-Learn.

1 Introduction

Team formation is a challenging task especially for human resource managers in today's competitive world. Companies and public bodies select individuals to work towards a predefined set of objectives or goals. But usually, such selection is done randomly. Random selection often leads to a situation wherein, the team may not perform optimally as needed by the organization. There is an uncertainty over the cohesiveness of the team. If there is an immediate requirement or requirements, a data-driven approach becomes a necessity. Valid data for the team can be obtained from various social networking sites. In these social networking sites, we receive data about an individual's past performance and collaboration efforts.

By using standard Machine Learning techniques, both supervised unsupervised we can find a certain pattern in this data. Based on their skills, past collaborations, individual performance and many such similar features their future performance in a group working in particular skillset towards a predefined goal can be predicted. This is the basic idea of what we are working on. In our case, the social Network is IMDB and the individuals are actors and directors. The team in our case is the movie. Performance is the average rating received by the movie from critics.

2 Related Work and Motivation

Previous efforts in this field have been done in [4] and [5]. [4] uses The MinAggrSol algorithm, which transforms the social network into a simpler graph G, and then it greedily picks nodes from G based on a utility measure, until a feasible team is found. [5] discuss Bipartite Graph Spectral Co-clustering to solve the problem of Team Prediction. We used certain concepts like Jaccard Distance in [5] in our approach below.

Previous works mentioned here generally use graph-based techniques along with greedy strategies. We used standard ML techniques like Single Layer Neural Network, Multi-Layer Neural Network, and Gaussian Naive Bayes. We have seen Neural Networks used in a variety of classification and regression tasks like in [6]. We used the concept of both single layer and Multi-Layer Neural Network for our problem of team formulation. We also implemented Gaussian Naive Bayes mentioned in [7].

3 Problem Statement

Given the dataset consisting of Movie IDs, Actors, Directors, Genres of the Movies, predict the performance of the team formed by actors and directors by classifying it in Movie Rating classes by considering following factors:

1. Individual Skills:
   Consider Individual skills in specific genres as well as overall individual skills. Skills are measured in terms of Movie Ratings on the scale of 1 to 100.

2. Past Collaboration:
   Consider collaboration amongst all possible combinations of individuals by considering performance of the movies in which they have worked together.

4 Process Flow

The process flow used in our implementation is as shown in Figure 1.

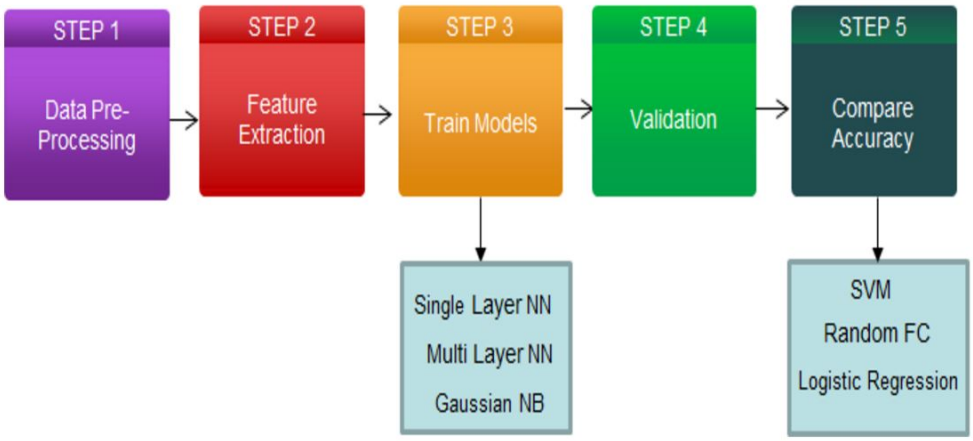

Figure 1: Process Flow used in our problem

The first and major step in any machine learning problem is the Data pre-processing. Once the data is processed, the selection of features for the classifier plays a major role as the performance of the classifier will be directly affected by the features selected.

After the selection of the features, the classifiers are trained using these input features and the class labels and are used for testing. In our project, we implemented Gaussian Naive Bayes, Single Layer Neural Network and Multi-Layer Neural Network to predict the performance of the team.

In the validation step, we have used 10-fold cross validation for validating the classifiers prediction accuracy.

The prediction accuracy of our implemented classifiers is compared across other classifiers like SVM, Random Forest, Logistic Regression, Decision Trees and Stochastic Gradient Descent.

5 Dataset

The dataset used in this project is the hetrec2011-movielens-2k dataset published by Grouplens Research Group. (http://www.grouplens.org) It links the movies of MovieLens dataset with their corresponding web pages at Internet Movie Database (IMDb) and Rotten Tomatoes movie review systems.

This dataset contains

- 10197 movies
- 4060 directors
- 95321 actors
- 20 genres and
- rated by 2113 users.

Each movie has on average 22 actors but since the lead actors in a movie contribute the most to the success of the movie, we considered only the top 5 ranked actors. Rankings of the actors can be obtained from the dataset. To get the rating of a particular movie, average of all the users who have rated the movie is considered.

The movie rating is scaled between 1 to 100. We divide this scale into 5 classes ranging from 1 to 5. Class 1 represents the lowest performance while class 5 represents the highest performance. The team is limited to up to 5 Actors, one director and any number of Genres up to 20.

6 Data Pre-processing and Feature Extraction

6.1 Data Pre-processing

The dataset sourced from MovieLens site has some inconsistencies with some special characters.

The initial step of the pre-processing scripts is to check the existence of any special characters in the underlying data and remove from the dataset. There are cases where some of the values are missing and we used additional parameters like mean and standard deviation to calculate the value of the missing fields.

Once this is done, a next level of pre-processing scripts will check for the existence of any potential outliers in the dataset and remove them accordingly. The final stages of the pre-processing will be to inspect all information and check the data available is consistent, reliable and complete.

The database used is MYSQL to load the data into tables from the files. All the tables are connected by the primary key 'MovieID'. The database schema implemented is as shown in Figure 2.

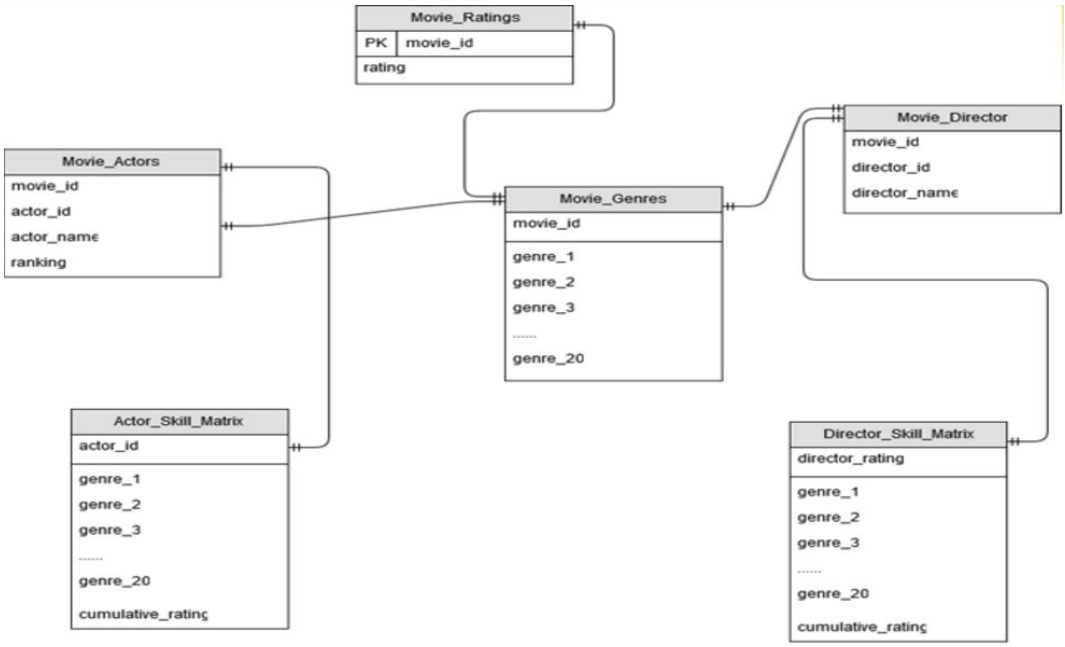

Figure 2: Tables loaded in MYSQL database

6.2 Feature Engineering

After the data pre-processing, further processing of the data is done to bring it to required data form. The feature engineering is done on this data to obtain 4 main features:

1. **Similarity Matrix** - Here the similarity between all pairs of individuals considered for the movie is found out and aggregated. Since, greater the similarity between the people,

greater will be the coercivity between the people and in turn a greater rating. It is calculated using the cosine distance which is as follows:

$$Similarity = cos(\theta) = \frac{A.B}{||A||||B||}$$

2. **Overall Quality** - Here the overall rating for every individual of the movie is found out and aggregated. Greater the individuals rating, greater will be the chance of the movie's rating to be high and the converse. It is calculated as follows:

$$OverallRating(P) = \frac{\sum_i rating_i(P)}{n(P)}$$

3. **Individual Quality** - Here the total genre wise rating for every individual of the movie is found out(based on the genre of the movie) and aggregated. The individual might be excellent at a particular genre but not so proficient in another, and this can be captured through this feature. It is calculated as follows:

$$Genrewiserating(P, genre) = \frac{\sum_i rating_i(P,genre)}{n_{genre}(P)}$$

4. **Collaboration Matrix** - Here the number of prior collaborations between all pairs of individuals considered for the movie is found out and aggregated. Greater the number of prior collaboration between the people, greater the chances of prior sucess and greater understanding between them. It is calculated using the jaccard distance which is as follows:

$$Distance(A, B) = 1 - \frac{||A \cap B||}{||A \cup B||}$$

7 Classifiers

The Classifiers implemented are as follows:

1. Gaussian Naive Bayes
2. Single layer Neural Network
3. Multi-layer Neural Network

7.1 Gaussian Naive Bayes

Naive Bayes classifiers belongs to the class of simple probabilistic classifiers based on Bayes rule application assuming strong conditional independence among the features. It is a generative classifier i.e., to find P(label | data) we first find P(data | label).

Naive Bayes classifier is highly scalable and the number of parameters is directly proportional to the number of features. The underlying assumption in Naive Bayes classifier is the value of a particular feature is independent of the value of any other feature, given the class variable, i.e., the features are conditionally independent of each other given class.

In our implementation, we have four input feature vectors and five class labels indicating the predicted rating.

In Gaussian Naive Bayes algorithm, it is assumed that the underlying distribution of the data is Gaussian.

7.1.1 Training

The posterior probability of a sample is given by Bayes rule,

$$P(Y = i|X) = \frac{P(X|Y=i)P(Y=i)}{P(X)}$$

Here, we need to calculate two components i.e, $P(X|Y = i)$ and P(Y=i).

P(class) is given by P(Y=i) = $\frac{Numberofsamplesinclass'i'}{Totalnumberofsamplesinthetrainingset}$

As the underlying distribution of the data is assumed to be Gaussian, our goal is to find the mean and variance of all the four features given class.

7.1.2 Testing

Using the mean and variance parameters found in training, we find the probability of a test sample given class 'j' using the formula,

$$P(Y = j|X) = \sum_{i=1}^{4} log(P(feature = i|class = j)) + log(P(class = j))$$

where,

$$P(feature = i|class = j) = N(\mu_{ij}, \sigma_{ij}^2)$$

In our implementation, the output vector 'y' in range of 0-100 is rearranged as shown below to support multi class labels (5 in our case):

$$if\ y[i] \leq 20\ \Rightarrow\ class\ 1\ \Rightarrow\ y[i][5]\ =\ [1,0,0,0,0]$$

$$if\ y[i] > 20\ \&\ y[i] \leq 40\ \Rightarrow\ class\ 2\ \Rightarrow\ y[i][5] = [0,1,0,0,0]$$

$$if\ y[i] > 40\ \&\ y[i] \leq 60\ \Rightarrow\ class\ 3\ \Rightarrow\ y[i][5] = [0,0,1,0,0]$$

$$if\ y[i] > 60\ \&\ y[i] \leq 80\ \Rightarrow\ class\ 4 \Rightarrow\ y[i][5] = [0,0,0,1,0]$$

$$if\ y[i] > 80\ \Rightarrow\ class\ 5\ \Rightarrow\ y[i][5] = [0,0,0,0,1]$$

The step-by-step algorithm implemented is:

Algorithm 1 Gaussian Naive Bayes Algorithm

1: Find the probability of each class label.
2: Find the mean m_{ij} of each feature vector 'i' given each class 'j'
3: Find the variance v_{ij} of each feature vector 'i' given each class 'j'
4: Calculate the maximum likelihood estimate of a feature belonging to a class using the equation mentioned in section 6.1.2.
5: The input feature should be assigned to the class with highest probability found using step 4.

Finally, the sample is classified to a class that has the maximum probability. The predicted class is compared with the original class and the prediction accuracy of the classifier is calculated.

7.2 Single Layer Neural Network

The basic structure of Artificial Neural Network is a Single Layer Neural Network, also known as single Layer Perceptron. It consists of only 2 layers, one input layer and one output layer. There are no hidden layers in Single Layer Perceptron. The input layer contains nodes equal to the number of features in the feature vector plus one. The extra input to the neural network represents bias. The output layer can have multiple nodes depending on the data and the requirement. As seen in the Figure 3, each input node is connected to the output node. If there are multiple output nodes, each input node is connected to each output node as seen in the Figure 4. Each connection has a weight associated with it. The output node takes the weighted sum of all the inputs. The final output is calculated by applying activation function to this weighted sum. There exist many activation functions like linear, TanH, Rectified linear unit (ReLU) etc.

Since our data set contains four features, we have used single layer neural network with total of five input nodes (One for bias) and one output node. We have used linear activation function as it gave the best accuracy for the given data. The algorithm of this implementation can be found in Algorithm 2. The output i.e. the class label is given by calculating the weighted sum of inputs and then applying the linear function as shown below:

$$D = w_0 + \sum_{i=1}^{N} w_i x_i$$

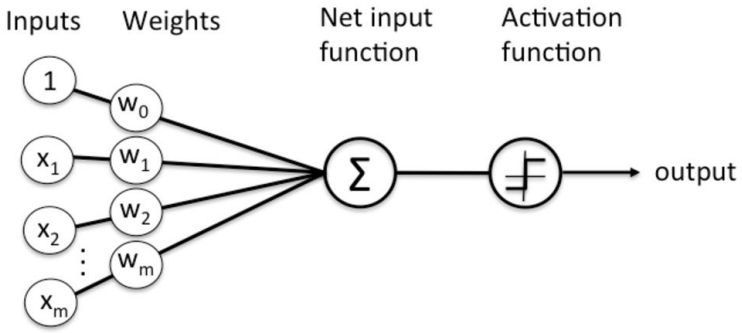

Figure 3: Single Layer Neural Network with single output node

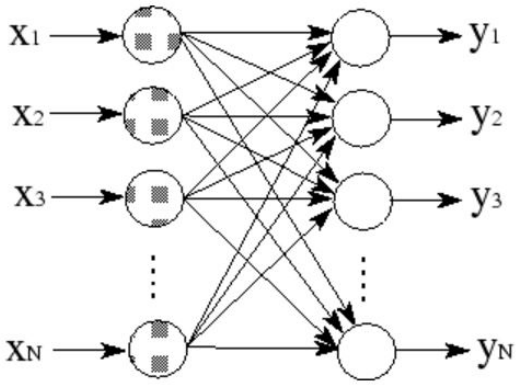

Figure 4: Single Layer Neural Network with multiple output nodes

The linear function used to determine the class label of the output D is as follows:

$$\text{linearFunc(D)} = \begin{cases} 0 <= D <= 20, & Class1 \\ 20 < D <= 40, & Class2 \\ 40 < D <= 60, & Class3 \\ 60 < D <= 80, & Class4 \\ 80 < D <= 100, & Class5 \end{cases}$$

We start with some initial set of weights and apply Back propagation algorithm using Gradient Descent as shown below:

$$w_i = w_i - \eta \Delta_{w_i} E$$

The optimal set of weights are learned in this process. The weights updating process is continued until the mean squared error becomes less than preset threshold. Mean Squared Error is given by the equation as follows:

$$E = \frac{1}{2N}(D-d)^2$$

Algorithm 2 Single Layer Neural Network Algorithm

1: Initialize the Weight Vector W = { w_0, w_1,...., w_5 }
2: Calculate the class label D.
3: Calculate mean squared error E.
4: Update the weights using Gradient Descent.
5: Repeat steps 1 to 4 until the average error is below error Threshold or for specified number of iterations.

7.3 Multi-Layer Neural Network

A Multi-Layer Neural Network is a feed-forward artificial neural network model that maps sets of input data to a set of appropriate outputs. It consists of multiple layers of nodes in a directed graph. Each layer is fully connected to the next immediate layer. Except the input nodes, each node is a processing element with a nonlinear activation function. A supervised learning technique called back-propagation is used for training the network by updating the weight vectors used in each layer of the network.

The two main Activation functions used are,

1. tanh activation function defined as

$$y(v_i) = tanh(v_i)$$

2. sigmoid activation function defined as

$$y(v_i) = (1 + e^{-v_i})^{-1}$$

The first activation function is a hyperbolic tangent whose value ranges from -1 to 1. The second activation function is a sigmoid whose values ranges from 0 to 1. Both of these activation functions are similar in shape as shown in figures 5 and 6.

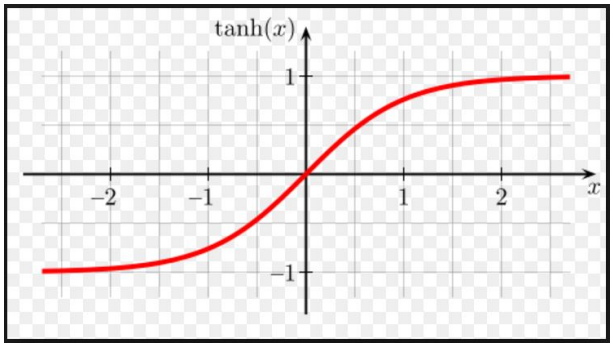

Figure 5: Tanh Function

The basic layers of Multi-layer neural network as shown in figure 7. include

1. Input Layer
2. Hidden Layer
3. Output Layer

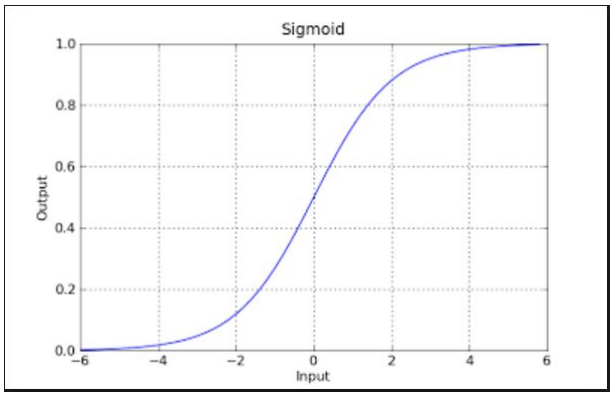

Figure 6: Sigmoid Function

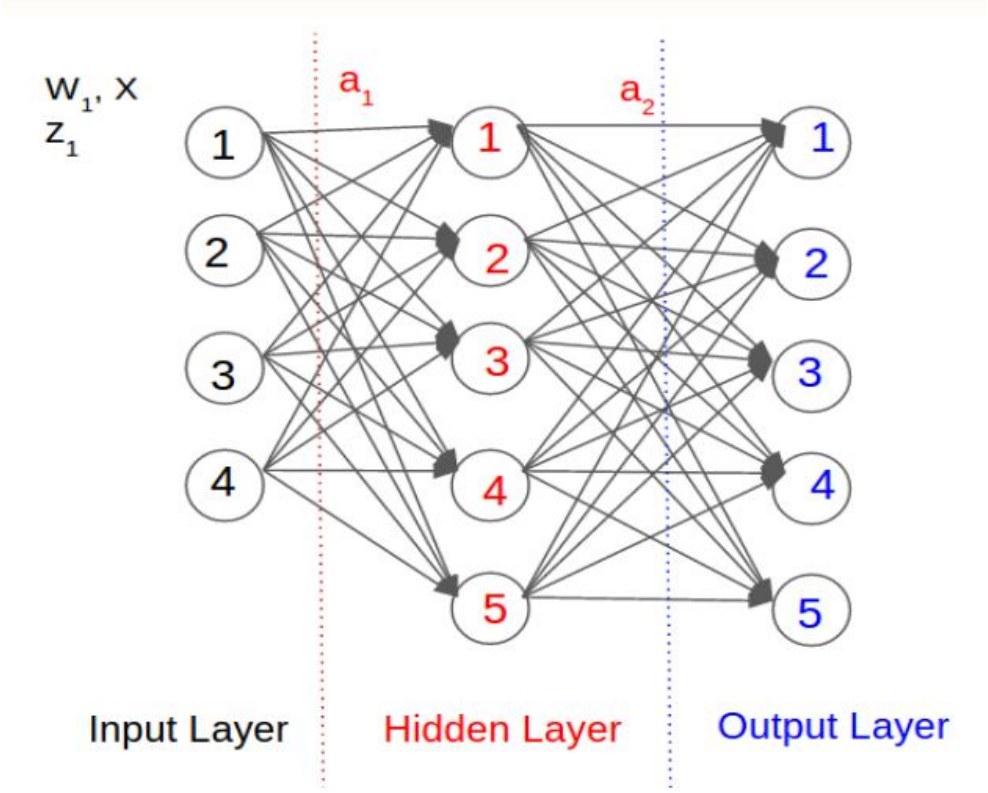

Figure 7: Multi-layer Neural Network

W_1 denotes the weight vector for input layer and W_2 denotes the weight vector for hidden layer. a_1 and a_2 denotes the activation functions before and after the non-linear transformation from the hidden layer.

The number of hidden layers and the number of nodes in each hidden layer can be varied. In our implementation, we have one hidden layer with 7 nodes, input layer with 4 nodes(bias of 1 is added to the input vector beforehand) and output layer with 5 nodes. In our implementation, the output vector 'y' in range of 0-100 is rearranged as shown below to support multi class labels (5 in our case):

$$if\ y[i] \leq 20\ \Rightarrow\ class\ 1\ \Rightarrow\ y[i][5]\ =\ [1,0,0,0,0]$$

$$if\ y[i] > 20\ \&\ y[i] \leq 40\ \Rightarrow\ class\ 2\ \Rightarrow\ y[i][5] = [0,1,0,0,0]$$

$$if\ y[i] > 40\ \&\ y[i] \leq 60\ \Rightarrow\ class\ 3\ \Rightarrow\ y[i][5] = [0,0,1,0,0]$$

$$if\ y[i] > 60\ \&\ y[i] \leq 80\ \Rightarrow\ class\ 4 \Rightarrow\ y[i][5] = [0,0,0,1,0]$$

$$if\ y[i] > 80\ \Rightarrow\ class\ 5\ \Rightarrow\ y[i][5] = [0,0,0,0,1]$$

7.3.1 Training

Initialize the weight vectors used in input and hidden layers. The training is done in two steps:
1. Forward Propagation
In our implementation, 'tanh' activation function is used.
Here, the final probability of class label given data is found using the formula

$$P(class|data) = tanh(W_2(tanh(W_1X))$$

The output predicted class will be non-linear transformation of the input and hidden layers.
2. Backward Propagation The weight vectors W_2 and W_1 are updated using back propagation gradient descent algorithm given below:

Algorithm 3 Back Propagation Gradient Descent Algorithm

1: Calculate the mean square error 'E' between predicted and actual class label.
2: Calculate the partial derivative of 'E' w.r.t W_2.
3: Update the weight vector W_2 using gradient descent.
4: Calculate the partial derivative of 'E' w.r.t W_1.
5: Update the weight vector W_1 using gradient descent.
6: Repeat the steps 2-5 for 5000 iterations to find the local minima.

The mean square error and weights updating using gradient descent is implemented using the same formulae mentioned in single layer neural network.

7.3.2 Testing

The class of the test data is predicted using the trained classifier. The predicted class is compared with the test data's original class and the prediction accuracy of the classifier is calculated.

7.4 Implementation of other Classifiers for performance comparison

To better understand which classifier may best suit the data at hand, we implemented some classifiers using a python library, Scikit-learn. The classifiers implemented are listed as follows:

- Support Vector Machines
- Logistic Regression
- Random Forests
- Decision Trees
- Stochastic Gradient Descent

Once we tested these classifiers we compared the accuracy we got with our implemented classifiers. Detailed results has been provided in section 9.2.

7.4.1 Pseudo-Code

Our Pseudo-Code for this module is as given below.

Algorithm 4 Multiple Classifier Accuracy Comparison

1: Loop through Feature data and pre-process the data labels class conditionally as done in implemented models.
2: Generate a New array of labels from step 1.
3: For all the classifiers to be tested loop follow through step 4-6.
4: Initialize the SciKit Learn Classifier and fit the data.
5: Test the classifier with 10 fold cross Validation and append it's accuracy to the array. Array was initialized in the first run.
6: Plot the final graph by appending the accuracy of implemented classifiers with accuracy array generated from step 5.

8 Cross Validation

Cross-validation is a validation technique to assess how the results of a statistical analysis generalizes to unseen data. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (testing dataset).

The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation dataset), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem.

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

One of the main reasons for using cross validation is to overcome the insufficiency of testing data in conventional testing techniques that usually involve partitioning the data into training(70 percent) and testing sets(30 percent). Hence, it's a good way to prevent overfitting without the requirement for a large testing set.

Algorithm 5 Cross Validation Algorithm

1: Segment the data into n equally sized sets(folds F_1)..F_n).
2: Run n experiments using F_1) as Test set use the other n-1 Folds together as the Training Set.
3: Select the best value of the error (or the accuracy) from all the runs for a given fold.
4: Repeat the experiment by selecting other folds F_2)..F_n) as test sets and calculate the error/accuracy for each fold.
5: Calculate the average of the error/accuracy for all the folds and that is the result from the cross validation algorithm.

9 Results

9.1 Effects of Parameters on Classifier Performance

We validated the results of the model using 2 types of validation,

- 10 fold cross validation
- Validation using two third data as training and one third as testing data

We experimented with different values of parameters involved to understand their effect on the classifier performance. We varied the values of learning rates and number of Epochs used in the Single Layer Neural Network and Multi-layer Neural Network and recorded the results.

Some of the Multi-layer Neural Network results are presented below. As observed from Figure 8, We can see that for initial 1000 iterations, the accuracy of the Multi-layer Neural Network classifier increased exponentially.After 1000 iterations, it decreased for a number of iterations and became almost stable after 4000 iterations. This decrease in accuracy indicates that more number of iterations caused the decision boundary to overfit the data. The decision boundary is complex and tries to correctly classify the training data but fails to classify the unseen data correctly. And hence, fails to generalize the model. Thus, after 4000 iterations there was no gain in the accuracy.

Learning rate is also one of the most important factors in deciding the performance of the classifier and an optimal learning rate ensures faster convergence of the algorithm with the best accuracy. Too high or too low learning rate can result in poor classifier performance and more learning time.

As observed in Figure 9, 5 x 10-7 proved out to be the optimal learning rate and 4800 epochs to be optimal number of epochs for our Single Layer Neural Network classifier.

The main aim of the Neural Net is to minimize the loss of the model. We calculate loss in terms of Mean Squared Error. As we have used gradient descent in Back Propagation algorithm, we can note the minimization of loss by observing the orange curve.

We observed that after 4800 iterations, there is no significant decrease in loss which means the algorithm should converge there. The blue curve associated with 9 x 10-7 learning rate demonstrates the descent in loss till 2500 iterations but we can observe a gain in loss in later iterations. This curve effectively demonstrates the effect of too high learning rate on the loss. Too low learning rate can cause loss improvements to be linear.

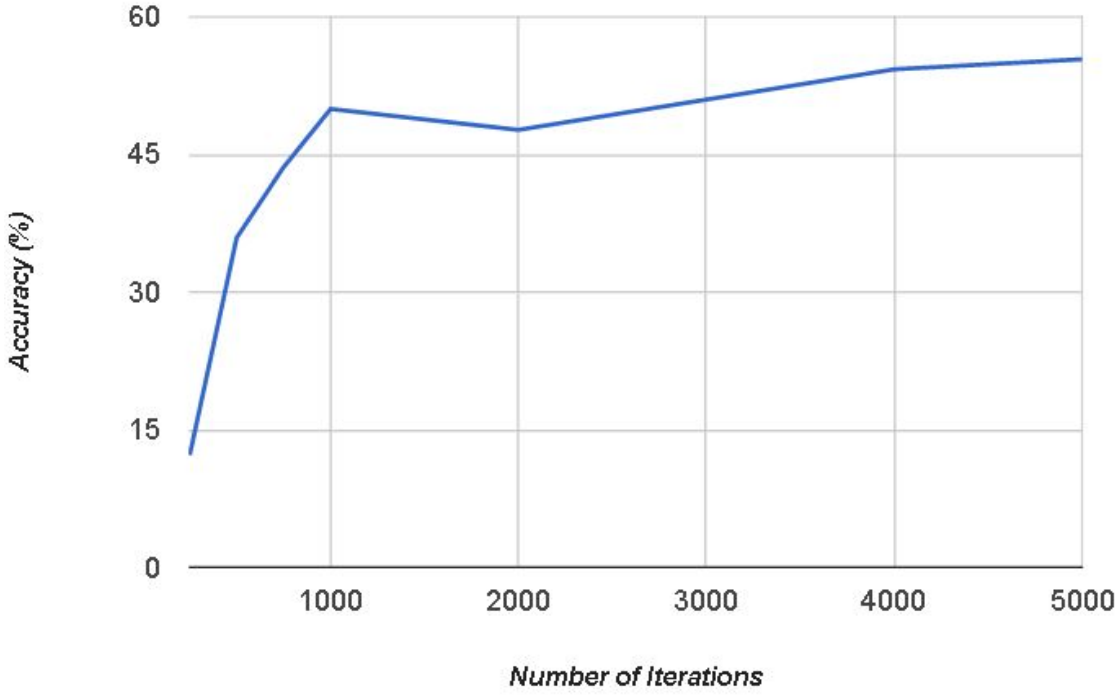

Figure 8: Effect of number of iterations on prediction accuracy of Multi-layer Neural Network

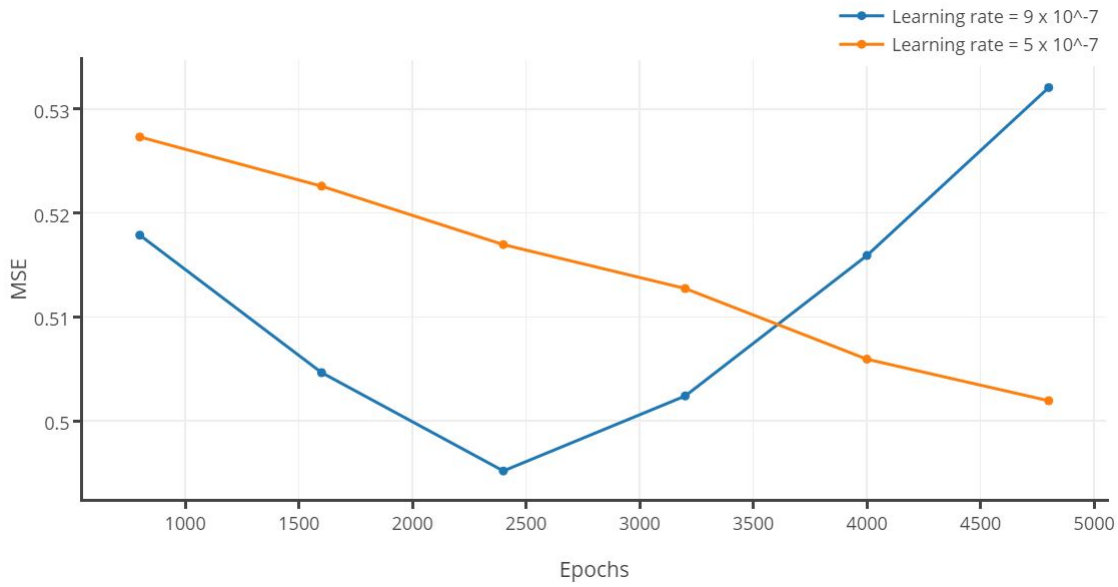

Figure 9: Effect of Learning Rate on Gradient Descent

9.2 Classifiers Comparison

Amongst the models developed by us,

Gaussian Naive Bayes performed the best with 62.7 % accuracy,
Multi-layer neural network with 55.37 % and
Single Layer Neural Network with 55.15 % accuracy.

Gaussian Naive Bayes assumes conditional independence amongst the features and hence, does not get affected by irrelevant features in the data and hence, it performed better than Neural networks. But the same assumption might have affected the Gaussian Naive Bayes Performance since the two features in our data namely individual quality and overall quality of a individual are partially dependent on each other as the genre wise rating considered in individual quality is a subset of the overall quality.

Multi-layer Neural Network performed better than Single Layer. Single layer Neural Network do not perform well on non-linearly separable data. Each layer in a feed-forward multi-layer Neural Network adds its own level of non-linearity to the classifier which cannot be achieved by Single Layer Neural Network.

Single Layer, weighted inputs from each layer are only linearly combined, and hence cannot produce the non-linearity that can be seen through multiple layers. However, multi-layer neural net can output a complex non-linear decision boundary which correctly classifies the given data points.

To validate our models and to understand which models may perform better than our model, we implemented some other classifiers like

- Support Vector Machines
- Decision Trees
- Logistic Regression
- Random Forests and
- Stochastic Gradient Descent.

Amongst them, we observed that Support Vector Machines (SVM) and Random Forests performed the best for our data with 69 % and 70 % accuracy respectively.
Accuracies for other models can be found in Table 1. SVM performs well with noisy data and prevents of over fitting of data. It tries to maximize the margin of the decision boundary which results in reduced mis-classfication.

Algorithm	**Accuracy**
Gaussian Naive Bayes	62.7%
Single Layer Neural Network	55.15%
Multi-Layer Neural Network	55.37%
Logistic Regression	59 %
Support Vector Machine	69%
Decision Trees	66%
Random Forests	70%
Stochastic Gradient Descent	45%

Table 1: Comparison across studied approaches

NOTE: The accuracies mentioned for Single Layer and Multi Layer Neural Networks is different from those submitted in slides as we experimented with more number of iterations and number of nodes in hidden layer is also changed.

10 Conclusions and future work

10.1 Conclusions

1. SVM performed the best for the given dataset as it works well with noisy data as it maximizes the margin around decision boundary resulting into less misclassfication errors.%.

2. The conditional independence amongst the features avoids the effect of irrelevant features on the classifier performance.

3. Multi-layer Neural Network performs better than Single Layer as more number layers form complex decision boundary over non-linearly separable data.

4. Learning rate has a great impact on accuracy and learning time of the model. Too high or too little learning rate may lead to higher train time and lower accuracy as it may pass over the global minima.

5. By varying the number of nodes in the hidden layer of Multi-Layer Perceptron from 2 -10, it was observed that our classifier implemented performed the best for 7 nodes.

6. There exists an optimal number of iterations which give best accuracy as there is no significant gain in accuracy after that. It is important to get the algorithm to converge at the right time to get best performance and prevent overfitting of data.

10.2 Future Work

1. There exist many other factors like Budget, Music Director, Locations filmed which impact the performance of the movie greatly. These features can be used to predict the performance of the movie.

2. Since the number of layers in multi-layer neural network is an important factor in getting the optimal decision boundary. The accuracy of the Multi-Layer Neural Network model can be improvised by adding more number of hidden layers.

11 Contributions

The individual contributions are as follows:

Team Member	**Contribution Description**	**Contribution in percentage**
Shalmali Bhoir	- Single Layer Neural Network Implementation - Experimentation with various parameters of Algorithms - Analysis of Results - Presentation and Report of the corresponding part	16.66%
Aditya Prasad Mishra	- Feature Engineering - Classifier Comparison - Analysis of Results - Presentation and Report of the corresponding part	16.66%
Manikandan Anandan	- Feature Engineering - Data Pre-processing - Presentation and Report of the corresponding part	16.66%
Piyush Nolastname	- Feature Engineering - Cross Validation Code - Analysis of Results - Presentation and Report of the corresponding part	16.66%
Pragna Munagala	- Gaussian Naive Bayes Implementation - Multi Layer Neural Network Implementation - Experimentation with various parameters of Multi Layer Neural Network Algorithm - Presentation and Report of the corresponding part	16.66%
Rohit Balachandran Menon	- Feature Engineering - Data Pre-processing - Presentation and Report of the corresponding part	16.66%

Table 2: Individual Contributions

12 References

[1] en.wikipedia.org/wiki/Multilayer_perceptron

[2] ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

[3] en.wikipedia.org/wiki/Naive_Bayes_classifier

[4]Majumder, Anirban, Samik Datta, and K. V. M. Naidu. "Capacitated team formation problem on social networks." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012.

[5] Farhadi, Farnoush, et al. "Teamfinder: A co-clustering based framework for finding an effective team of experts in social networks." Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on. IEEE, 2012.

[6] Emanuelsson, Olof, Henrik Nielsen, and Gunnar Von Heijne. "ChloroP, a neural network-based method for predicting chloroplast transit peptides and their cleavage sites." Protein Science 8.5 (1999): 978-984.

[7] Ng, Andrew Y., and Michael I. Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes." Advances in neural information processing systems 2 (2002): 841-848.

[8] Trippi, Robert R., and Efraim Turban. Neural networks in finance and investing: Using artificial intelligence to improve real world performance. McGraw-Hill, Inc., 1992.

[9] Howard B. Demuth, Mark H. Beale, Orlando De Jess, and Martin T. Hagan. 2014. Neural Network Design (2nd ed.). Martin Hagan, , USA.

[10] en.wikipedia.org/wiki/Cross-validation_(statistics).

[11] www.gabormelli.com/RKB/Cross-Validation_Algorithm.