Assignment 6

This is the final report on your implementation, and it requires a short video from you (Section 6.10). Paste Assignments 1 - 5 at the head of this *as is*, except that in Assignment 5, respond to each of your facilitator’s comments within each of their comments. Again, retain these gray parts. Keep in mind the evaluation criteria (at the end) and that they are somewhat different for this last assignment.

For voluminous material, reference appendices (at the end). These will be read on an as-needed basis.

Excluding appendices and figures, this response should not exceed 6 pages—without prior permission. As you do this last assignment, keep in mind the discussion thread at *Discussion Board Forum: Ask Instructional Staff Thread: The Final* because working on your project is also preparation for the final exam.

# ASSIGNMENT 1: PROJECT PROPOSAL DRAFT

Please use this template. Retain the gray text. Your materials—in black 12-point Times New Roman—should not exceed 5 pages excluding references and figures. Note the evaluation criteria, and leave plenty of time for editing that best responds. We recognize that you may alter your plans as the term progresses. That is to be expected—your changes or substitution will fit with this growing document.

## 1.1 SUMMARY DESCRIPTION

One- or two-paragraph overall description of a proposed term project. You will be free to change this in future but we want you to start thinking this through as early as feasible because implementation can be time-consuming.

I would like to build a media recommendation system. It could be for books or TV Shows or Movies or Music, depending on the availabilities of datasets. I feel like recommendation systems are almost an everyday part of our life now, given that everyone uses services like Youtube, Netflix, Amazon daily. The goal of the project would be to build a Neural Network that processes User data based on their past likes and dislikes and recommends new material to the user.

## 1.2 I/O EXAMPLES

At least two specific examples of projected output for designated input. You will not be held to this—it is just explanatory at this point.

The Dataset would include 2 important aspects of data:

1. Information on the items: The ID of the item, the number of people who have accessed this item, and the average rating of the item
2. Information on the user: The ID of the user, a history of items the user has accessed, and the ratings that the user has given for each of these items

The input to the neural net would be the information about the user in a vectorized form, and the output would be the predicted ratings for all the items available in the dataset. These ratings will then be sorted to present to the user the top recommendations (i.e the items with the highest predicted rating.)

## 1.3 REQUIREMENTS

High-level requirements statement in 3 roughly equal numbered lists, organized by triage. Separate your requirements into thee approximately even categories using triage (select the two extreme categories—definite and nice-to-do—and then place the remainder in the middle category). State requirements in declarative language such as “The application will recognize numbers 0-9 from a 12 by 35 array of black-or-white pixels” (not “First I will build a neural net”—which is a procedure rather than a requirement). Be conservative with your “definite” requirements: capabilities that are the bare necessity to have an actual project but no more.

### 1.3.1 Definite Requirements (first priority)

To arrange the data in a manner that can be presented as input to the neural network

To predict the ratings for items as given by a user (should be feasible for each individual user) for the items that user has not accessed yet.

### 1.3.2 Not classified yet (second priority)

To also generate a neural network that can identify the most popular items in a recent time frame to show the user items in a different list which will be called the “Currently Popular items” list.

### 1.3.3 Nice-to-do (can dispense with if time does not allow; third priority)

To achieve both the above tasks, but also categorized by genre

## 1.4 HOW SUCCESS WILL BE ASSESSED

Explain, as specifically as possible (quantification is ideal) how success of the project should be assessed. An example is “80% successful recognition of a cat in random images.”

The dataset will be divided into training and testing dataset. The success of the system will be based on the accuracy of the system.

## 1.5 TECHNOLOGY EXPLANATION

Explain what two technologies you are seriously considering--and why you feel they apply. One may be emphasized as the implementation and the other as an alternative. If at all possible, show fragments of experimentation with these. An example is [here](https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.88592&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&co).

The model I want to implement for this project is a Neural Network. I feel like it fits because once the model is trained, the model is very fast to run. And all the data for this particular task already is, or can very easily be converted into numerical values.

If not Neural Nets, other classification algorithms like KNN or Decision Trees can also be considered. The approach to the project would be slightly different if these are used. The goal in this case would be to find similar users to the input user and recommend items that the other user liked, but this user has not accessed yet.

## 1.6 DATA SOURCES

Explain whether or not your project requires data. If so, describe were you will obtain it.

The dataset can be found on Kaggle. I’m not sure I will be using this particular one but an example dataset I found on the internet can be found [here.](https://archive.org/download/nf_prize_dataset.tar)

## 1.8 REFERENCES FOR PROPOSAL PHASE

Fill in and cite each of the following (e.g., “[2]“) within the text. References can include specific places in the notes and textbook.

# ASSIGNMENT 2: PROJECT PROPOSAL PLUS

## 2.0 WHAT’S CHANGED

Provide no more than a page of 12-point type explaining what has been changed or added since assignment 1. Include in this whether and how the material in module 2 influenced this, or refer to reading that you did in working on this assignment (#2).

<Your response replaces this>

1. Decided on a type of model: In a similar project I did on my own a while ago, I used collaborative filtering to generate results, which is mainly using math and cosine similarity to generate similar users. In doing research, I found a blog post[1] about implementing a flavor of collaborative filtering in Keras[2]. I will mostly be using this for my project
2. I found a library “Beautiful Soup” which is basically a web-scraper. I want to use this or some other web scraping library to incorporate the posters of the movies the system recommends. (This is a new addition, but it is a stretch goal. I will be implementing this only if I complete the basic code in time.)
3. I have decided on a dataset. It is the MovieLens dataset.[3]

## 2.1 SUMMARY DESCRIPTION, VERSION 2

One- or two-paragraph overall description of your proposed term project. Giving your application a name is usually a good idea.

I would like to build a Movie recommendation system. I feel like recommendation systems are almost an everyday part of our life now, given that everyone uses services like Youtube, Netflix, Amazon daily. The goal of the project would be to build a Neural Network that processes User data based on their past likes and dislikes and recommends new movies to the user.

## 2.2 I/O EXAMPLES, VERSION 2

At least two specific examples of projected output for designated input. You will not be held to this—it is just explanatory at this point.

1. Input to the system will be the ID of a user  
   eg: UserID = 200713

Output given by the system  
: A Ranked list of movie recommendations.

1. Input to the system will be the ID of a user and the ID of a movie  
   eg: UserID = 200713, MovieID = 12345  
   Output given by the system  
   : Predicted rating as given by the user for the Movie  
   eg: UserID = 200713, MovieID = 12345 => Predicted Rating: 3.37

## 2.3 REQUIREMENTS, VERSION 2

High-level functional requirements statement in two roughly equal numbered lists, organized by triage. Separate your requirements into two approximately even categories (select modest “definite” requirements, otherwise “nice-to-do”). This organization allows you to first attain readily do-able goals without getting bogged down, and then move on to other goals if you can. State requirements in declarative language such as “(Recognize 0-9): The application will recognize numbers 0-9 from a 12 by 35 array of black-or-white pixels” (not “First I will build a neural net”). Giving each requirement a label (e.g., “(Recognize 0-9):”) helps with clarity and readability.

A ratings dataframe containing details about the ratings for the movies in the dataset.  
A Users dataframe containing details about the ratings for the users in the dataset.  
A Movies dataframe containing details about the ratings for the movies in the dataset.

### Definite Requirements (first priority)

Predict ratings for unseen movies by the user and a) sort and predict the top recommended movies and b) predict the rating given a movie, user pair

### Nice-to-do (second priority)

* Incorporate movie posters in the system as described in section 2.0
* Incorporate genre based recommendations .

## 2.4 V2: HOW SUCCESS WILL BE ASSESSED

Explain, as specifically as possible (quantification is ideal) how success of the project should be assessed. We don’t want aimless projects of the form “I’ll play around with X until time runs out” because they are less motivational and because you learn less.

I will use Mean Squared Error on the test dataset to evaluate success of the system.

## 2.5 V2 TECHNOLOGY EXPLANATION

Explain what two technologies you intend to use--and why you feel they apply to your particular project. One of the two may be emphasized as the implementation and the other as an alternative or as a complement—discussed but not implemented if need be. If possible, show fragments of code execution. For example, if you are using TensorFlow, show that you have run some code. This can be simple—we just want you to break the ice with implementation.

I intend to use Neural Nets to implement a flavor of Collaborative Filtering to implement recommendation. Collaborative Filtering is one of the most effective recommendation algorithms without incorporating machine learning. This is the type of problem Neural Nets would naturally be good at solving. Combining both these techniques will most likely give great results.   
I will use BeautifulSoup or some other web scraping tool to get movie posters.

## 2.6 V2 DATA SOURCES

Explain whether or not your project requires data. If so, describe were you will obtain it. Be careful about this because you won’t have a project if it needs data and you have to spend too much time hunting and gathering it.

My dataset is publicly available at : <https://grouplens.org/datasets/movielens/>

## 2.8 REFERENCES FOR PROPOSAL V2

Fill in, and also cite each of the following (e.g., “[2]“) within the text. References can include specific places in the notes and textbook. You are free to include references from the prior assignment version.

[1] <http://www.fenris.org/2016/03/07/index-html>

[2] <https://keras.io/>

[3] <https://grouplens.org/datasets/movielens/>

.

# Assignment 3: Project Design, version 1

Keep in mind the evaluation matrix at the end as you do the work and use it to guide what you submit. Use no more than 6 pages of 12-point text excluding figures. You may include as many appendices as you wish for reference. Parts of these may be read as needed.

## 3.1 Final Requirements

List your final requirements, numbering them in the form DiX and NiX where:

D/N means “Definite” / “Nice to do” (two categories, not three)

i = 1, 2, 3, …

X=L and the goal is a *learning* goal – or – X=F and the goal is *functional*

You will reference these numbered requirements in the rest of the term, when you will be asked to show what the project accomplished.

D1-F: Build a Collaborative Filtering Neural Network that is able to predict the rating for a movie as given by a user.   
D2-L: Play around with various parameters of the Neural Network such as the number of Embeddings, Layers, Optimizers, Epochs, Learning Rates to see how it affects the performance of the Model.   
N1-F: Build a second Model that also takes into account the genre of the movie as a factor in recommending items to a user.   
N2-F: Use a web scraper to attach movie posters to the results while displaying recommendations, to improve the experience.

## 3.2 Design and Theory

Describe the design of your proposed system. Use annotated diagrams. Explain the theory behind your design. Explain how the two technologies will interface or compare. The reader should understand how you plan to fit the pieces together. Show this at a high level, as well as providing as much detail as you can at this point. Include at least one (meaningful) figure, for example, using boxes and arrows showing data flow.

Here is a high level design for the flow of the program and the model:

1. Pre-process the data:
   1. This includes looking at the shape of the data, removing inconsistencies and garbage values etc.
2. Split the data into train and test:
   1. This will most likely be a 80-20 or 70-30 split, using a library like sklearn.train\_test\_split[1]
3. Defining the Model
   1. Creating separate embeddings[2] for the Movies matrix and the Users Matrix
   2. Merging the embedded matrices using a dot product
   3. Applying several other Neural Network Layers[3] such a Flattening, Dense Layers, Dropout Layers after the dot product of the matrices. (In a combination that gives the best results, as I will be experimenting with this)
4. Deciding parameters such as Optimizers, Epochs, etc
5. Train the model
6. Use the trained model for predictions

## 3.3 Tools

Describe the tool(s) you will probably use, or explain why you will build from scratch. It is OK if you say "I will use tool 1 or tool 2." Support the fact that you have reasonably investigated and tried out tools. Explain your choice. Show samples that make you and us reasonably confident of your choices. Show that you understand how the tools work.

1. Jupyter Notebook[4] : Will be used for implementation
2. Keras :   
   Will be used to load and train the actual model. I considered using Pytorch as well but Keras is easier to understand in terms of writing the code as well as readability.
3. Beautiful-Soup[5]: Will be used to get movie posters
4. Matplotlib[6]: Will be using this library to plot the loss, accuracy, and other metrics for the model.

## 3.4 Risk Retirement

Identify and prioritize the 5 top risks in carrying out the project. Try as best you can to retire (set to rest) the top two by the time you submit this, by means of experiments, prototypes, or work-arounds. Explain how you did this. Explain how you will retire the remaining ricks in advance.

1. Time constraint: Since the movie dataset is very large, and I hope to try out various configurations of the model to figure out which one gives me the best results, training and testing each model might be difficult in a short amount of time.   
   Workaround: I can use Google Cloud platforms to run multiple models on multiple machines simultaneously to get results faster than running them sequentially on my local machine.
2. Size of the dataset: Since the dataset is pretty large, it is possible that Pandas may not be able to process all of it at once, especially for the pre-processing stage because I will need to actually be able to head or tail the dataset at this stage to see some samples.   
   Workaround: For pre-processing I can use Pyspark to be able to deal with a larger dataset. Once the dataset is preprocessed I can train and test the model on the cloud, because at that stage I will not need to actually see the data.
3. Deciding on accuracy: Currently I am planning on using MSE to evaluate the performance of my model but I cannot guarantee that it is the best measure of evaluating my model.   
   Workaround: I will plan to run my model with two or three different loss functions blindly to see which gives the best results.

## 3.5 Schedule

Explain in outline the steps you intend to take to carry out the project. Show the completion of the stages. Include a schedule, as detailed as can be reasonably foreseen.

Week 4: Complete Pre-processing  
Week 5: Complete the first Model and the tuning of some parameters  
Week 6: Complete the tuning of the remaining parameters and decide on the best set of parameters. Implement the web-scraper.

## 3.6 References

Add to your references. Instructions as before.

[1] [https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html [2](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html%20%5b2)] <https://keras.io/layers/embeddings/>  
[3] <https://keras.io/layers/about-keras-layers/>  
[4] <https://jupyter.org/>  
[5] <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>  
[6] <https://matplotlib.org/>

# Assignment 4: Project Design Plus, version 2

Keep in mind the evaluation matrix at the end as you do the work and use it to guide what you submit. Use no more than 6 pages of 12-point text excluding figures. You may include as many appendices as you wish for reference. Parts of these may be read as needed. This revision is your final view of the design prior to implementation (though you may still change it when you implement).

(Note that a lot of stuff is same as the last iteration because not a lot has changed in the design part, I spent most of the week furthering my implementation and so far everything is on track.)

## 4.1 V2 Final Requirements

D1-F: Build a Collaborative Filtering Neural Network that is able to predict the rating for a movie as given by a user.   
D2-L: Play around with various parameters of the Neural Network such as the number of Embeddings, Layers, Optimizers, Epochs, Learning Rates to see how it affects the performance of the Model.   
N1-F: Build a second Model that also takes into account the genre of the movie as a factor in recommending items to a user.   
N2-F: Use a web scraper to attach movie posters to the results while displaying recommendations, to improve the experience.   
N3-L: Look at the scalability of the code. The movielens dataset is available in various sizes. Once the project is perfected on a smaller scale on the local machine, try to run it on larger sizes with more computing power on the cloud.

## 4.2 V2 Design and Theory

Here is a high level design for the flow of the program and the model:

1. Pre-process the data:
   1. This includes looking at the shape of the data, removing inconsistencies and garbage values etc.
2. Split the data into train and test:
   1. This will most likely be a 80-20 or 70-30 split, using a library like sklearn.train\_test\_split[1]
3. Defining the Model
   1. Creating separate embeddings[2] for the Movies matrix and the Users Matrix
   2. Merging the embedded matrices using a dot product
   3. Applying several other Neural Network Layers[3] such a Flattening, Dense Layers, Dropout Layers after the dot product of the matrices. (In a combination that gives the best results, as I will be experimenting with this)
4. Deciding parameters such as Optimizers, Epochs, etc
5. Train the model
6. Use the trained model for predictions

## 4.3 V2 Tools

Describe the tool(s) you will definitely use, or explain why you will build from scratch. ~~It is OK if you say "I will use tool 1 or tool 2."~~ Support the fact that you have reasonably investigated and tried out tools. Explain your choice. Show samples that make you and us reasonably confident of your choices. Show clearly that you understand how the tools work (in this, the next section will help too).

1. Jupyter Notebook[4] : Will be used for implementation
2. Keras :   
   Will be used to load and train the actual model. I considered using Pytorch as well but Keras is easier to understand in terms of writing the code as well as readability.
3. Beautiful-Soup[5]: Will be used to get movie posters
4. Matplotlib[6]: Will be using this library to plot the loss, accuracy, and other metrics for the model.

## 4.4 Implementation Fragments

Show enough *parts* of an implementation—or a simplified form of it—to convince yourself and the reader that you will have the implementation of the definite requirements completed on time. Try to make these part of the actual intended application, but they can be experimental or exploratory in nature. Your choices can coordinate with section 4.4 below. Cut and paste commented code below.

I have attached the implementation so far as a separate file in the assignment.

## 4.4 V2 Risk Retirement

Identify and prioritize the 5 top risks in carrying out the project. Try as best you can to retire the top ~~two~~ four by the time you submit this, by means of experiments, prototypes, or work-arounds. Explain how you did this. Explain how you will retire the remaining risks in advance. Update this from the version in Assignment 3.

1. Time constraint: Since the movie dataset is very large, and I hope to try out various configurations of the model to figure out which one gives me the best results, training and testing each model might be difficult in a short amount of time.   
   Workaround: I can use Google Cloud platforms to run multiple models on multiple machines simultaneously to get results faster than running them sequentially on my local machine.
2. Size of the dataset: Since the dataset is pretty large, it is possible that Pandas may not be able to process all of it at once, especially for the pre-processing stage because I will need to actually be able to head or tail the dataset at this stage to see some samples.   
   Workaround: For pre-processing I can use Pyspark to be able to deal with a larger dataset. Once the dataset is preprocessed I can train and test the model on the cloud, because at that stage I will not need to actually see the data.
3. Deciding on accuracy: Currently I am planning on using MSE to evaluate the performance of my model but I cannot guarantee that it is the best measure of evaluating my model.   
   Workaround: I will plan to run my model with two or three different loss functions blindly to see which gives the best results.

## 4.5 V2 Schedule

Explain in outline the updated steps you intend to take to carry out the project. Show the completion of the stages. Include a schedule, as detailed as can be reasonably foreseen.

Week 4: Complete Pre-processing and the first Model (ahead of time on this one as I originally planned on working on the model next week.)  
Week 5: Complete the tuning of the remaining parameters and decide on the best set of parameters. Implement the web-scraper.  
Week 6: Implement the web-scraper. Look at scalability(i.e run on larger dataset on the cloud)

## 4.6 V2 References

Add to your references. Instructions as above.

[1] [https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html [2](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html%20%5b2)] <https://keras.io/layers/embeddings/>  
[3] <https://keras.io/layers/about-keras-layers/>  
[4] <https://jupyter.org/>  
[5] <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>  
[6] <https://matplotlib.org/>

# Implementation v1 (version 1)

## 5.1 Summary v1

In a paragraph or two, summarize the outcome of your project functionally and learning-wise but avoid duplication with Section 5.3 below, so exclude details here.

My project is a movie recommendation system. As input data, I have a big Dataframe where each row is a unique rating that a user has given to a movie. I have used matrix factorization and neural networks to train a model for my recommendation system. Once the model is trained, the system can show the predicted rating for a <user, movie> pair or show the Top N Recommended movies for a input userID.

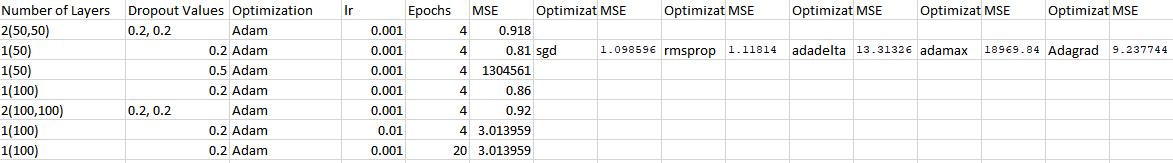
## 5.2. Report on Requirements v1

Explain the extent to which you accomplished each definite requirement "DiX" (X = F or L) *so far* as well as any other fulfilled requirements. For each, include 1-4 sentences and screenshot, as appropriate. Your effectiveness depends largely on how much you demonstrate that you learned, not necessarily that you implemented every requirement.

## D1-F: Build a Collaborative Filtering Neural Network that is able to predict the rating for a movie as given by a user.

Report: The Neural Network is fully functioning and has an MSE varying between 0.8 and 1.2 based off of the various parameter tunings I have been doing. I still have to write an interface function to the model, i.e write a simple function that a user can interact with which will call the model in the background to do the processing. I am not worried about this because it will take me around half an hour to do so, and I wanted to focus on parameter tuning first.

## D2-L: Play around with various parameters of the Neural Network such as the number of Embeddings, Layers, Optimizers, Epochs, Learning Rates to see how it affects the performance of the Model.

Report: I did a lot of parameter tuning, and I have built a table for the performance comparison of all of my “versions” of the Neural Network. Here I am judging the performance of my model based on the MSE value on the test data.   
  
  
I will go into detail about each parameter that I tuned and my observation regarding that.   
  
**1) Number of Layers**: The first layer of my model contains vectors that have the information about the users and the movies, and this is constant across all the variations of the models as this is essential to the functioning of the model. After that, I had either 1 or 2 densely connected layers to improve performance, and that is what the screenshot indicates. 2(50, 50) represents two dense layers having 50 nodes each. What I observed is that more densely connected layers do not necessarily mean better performance. After a point the model can overfit and it does not always make sense to have more layers because it does not guarantee a better result.   
**2) Dropout Values**: Pretty much all the reading I did on sample models and regarding Neural Networks had dropout layers after densely connected layers. It makes sense to have these because otherwise the model almost always overfits, giving a great validation result during the training but a definite drop in performance during testing. Here I do fully understand the problem that was discussed a lot during the course – “How to make Neural Networks better explain themselves.” One version of the model, which retained everything from its previous iteration (see 3rd row in the screenshot) except that the dropout value changed from 0.2 to 0.5. The MSE value went from 0.81 to 1304561. To me this makes no sense, and I can’t explain it, which is why I kind of wish that Neural Nets weren’t such a black box.   
**3) Optimization:** I didn’t change optimizers until pretty late in the tuning phase, because the one I was using by default is Adam, which in my experience will always give good results. So I ran optimizer changes only on the configuration that gave me the best results in the first round of testing (2nd row in the screenshot). As I had expected, Adam beat pretty much all the other optimizers Keras has to offer. I do know the theory and the math behind some of these optimizers (SGD, RMSprop) but not all, and again I couldn’t explain exactly why Adam was performing better compared to the others. It was just a matter of trial and error.   
**4) learning rate:** A lot of the Keras pages and Neural Net explanations had 0.001 as the learning rate so I decided to keep that for most of my testing. I know that a slower learning rate over a larger number of epochs is supposed to be the best combination but sometimes it can tend to overfit. 0.001 is a low enough number to allow the network to fully capture the features of the data without taking a super long time to train.   
**5) Epochs:** My first ever test was on 10 epochs (not included in the chart), and because Keras shows validation loss and validation MSE, I noticed that the best result always was around epoch 4 after which the model started to overfit. I’ve thus kept 4 epochs for most of my tests, and you can see that when I increased the number the MSE number took a hit.

## N1-F: Build a second Model that also takes into account the genre of the movie as a factor in recommending items to a user

Report: I feel like to some degree the implicit nature of the recommendation system would take the genre into account. However, I have not explicitly added this feature yet as my rating matrix did not have genres included, it would require me to go back and preprocess the data and do web scraping to maybe get the genre for a movie. It feels unlikely that I will be able to complete this in due time, so it is more than likely that I’ll drop this requirement.

## N2-F: Use a web scraper to attach movie posters to the results while displaying recommendations, to improve the experience.

Report: I have been doing some reading up regarding this, and the code for this would definitely reside in the interface function. It won’t take up much time and I am confident that I’ll be able to do it within the coming week.

## N3-L: Look at the scalability of the code. The movielens dataset is available in various sizes. Once the project is perfected on a smaller scale on the local machine, try to run it on larger sizes with more computing power on the cloud.

Report: I would consider this requirement as complete, because my code definitely works on the largest version of the dataset that MovieLens has to offer. I have been doing my parameter tuning on my local machine with the smaller dataset, but I plan to take my top 2 configurations and run them on the cloud on the larger dataset.

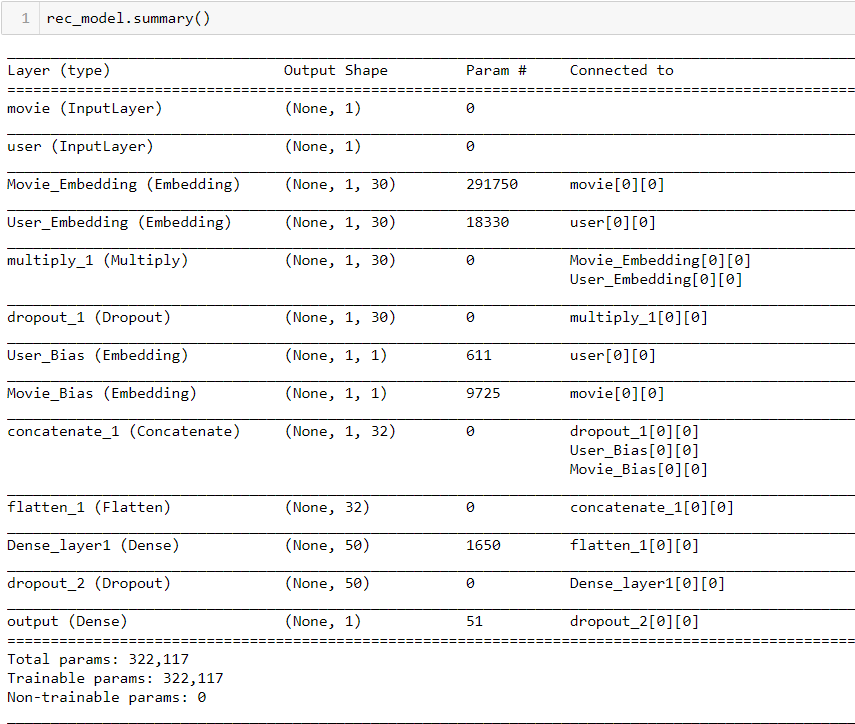
## 5.3 Report on Design v1

Describe the design that you have used so far. Indicate how, where, and why it has differed (thus far) from your planned design. Describe its advantages and its shortcomings. Include a description of how the technologies you explored (not the tools—those are described below) leveraged each other. Include at least one diagram.

1. Pre-process the data:
   1. This includes looking at the shape of the data, removing inconsistencies and garbage values etc.
2. Split the data into train and test:
   1. This will most likely be a 80-20 or 70-30 split, using a library like sklearn.train\_test\_split[1]
3. Defining the Model
   1. Creating separate embeddings[2] for the Movies matrix and the Users Matrix
   2. Merging the embedded matrices using a dot product
   3. Applying several other Neural Network Layers[3] such a Flattening, Dense Layers, Dropout Layers after the dot product of the matrices. (In a combination that gives the best results, as I will be experimenting with this)
4. Deciding parameters such as Optimizers, Epochs, etc
5. Train the model
6. Write an interface function that interacts with the user to take input such as a userId (to get top recommendations) or a <user, movie> pair (to get the predicted rating)
7. Implement the web scraper in the interface function to show movie posters along with the results.

This is the design I had submitted as a part of Assignment 4, and it remained unchanged over this week.

The internal design of my Neural Network is shown in the screenshot below.



The number of dense layers and the dropout values changed according to the parameter tuning process I explained earlier.

**Why use Embedding Layers?** As you can see in the model summary, we Multiply the user and the movie vectors as a part of Collaborative Filtering. This creates a pretty sparse matrix. Embedding layers help the network deal with sparse matrices better. What an embedding layer basically does is convert the input into lower dimensions that are easier to train and understand for the network. [1]

**Why use Dropout Layers?** Dropout layers are, as I explained earlier, essential to make sure that the model does not overfit. A dropout layer essentially turns off a certain percentage of Neurons during a forward or backward pass; essentially shutting down those parts of the network.   
  
**Why use Dropout Layers?** A dense layer is a fully connected layer where each neuron of the Dense Layer is connected to each neuron of the next layer. A densely connected layer provides learning features from all the combinations of the features of the previous layer, whereas a convolutional layer relies on consistent features with a small repetitive field. [2]

## 5.4 Tools v1

Describe the tool(s) that you are using. Show samples. Describe their advantages and their shortcomings. Limit: 1 page of 12-point Times New Roman.

Collaborative Filtering: This is a technique used in Recommendation Systems. This method focuses on finding “users like you” to make recommendations. The implementation involves creating a product of the user and movie matrices, which usually results in a pretty sparse matrix. I have used Neural Networks along with Collaborative Filtering but it can also be done in a purely mathematical sense without “learning algorithms”

Neural Networks: Since all of my data was numerical and neural nets work really well with numerical data it made sense to go with neural networks. The only disadvantage is that they’re somewhat of a blackbox. When I was tuning parameters there was a lot of trial and error without understanding why a particular change is better or worse. However, the results of the system are great.

# Implementation v2

## 6.1 Summary v2

In a paragraph or two, summarize the outcome of your project functionally and learning-wise but avoid duplication with Section 5.3 below (which is detailed). Underline edited sentences and additions from v1, if any.

My project is a Movie Recommendation System. I am using Neural Networks to implement this system. I am implementing the Network using the concept of Collaborative Filtering, which is incorporated in the Neural Network using Matrix Factorization.  
The input to my project will be three files, one file on user details, one file on movie details, and one file on rating details. The output of my project will be predictions of ratings as given by users.   
Once the model is trained, the system can show the predicted rating for a <user, movie> pair or show the Top N Recommended movies for a input userID.   
Through the implementation of this project I learned the usability, limitations, and the effect of hyperparameters of Neural Networks.

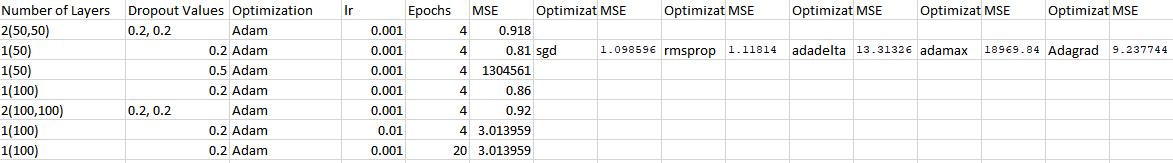
## 6.2. Report on Requirements v2

Explain the extent to which you accomplished each definite requirement "DiX" (X = F or L), as well as any other fulfilled requirements. For each, include 1-4 sentences and screenshot(s) (including of command-line text). Your effectiveness depends largely on how much you demonstrate that you learned, not necessarily that you implemented every requirement. Underline edited sentences and additions from v1, if any.

## D1-F: Build a Collaborative Filtering Neural Network that is able to predict the rating for a movie as given by a user.

Report: The Neural Network is fully functioning and has an MSE varying between 0.8 and 1.2 based off of the various parameter tunings I have been doing. I wrote an interface function which interacts with the model to get predictions and show them to the user in a presentable manner.

## D2-L: Play around with various parameters of the Neural Network such as the number of Embeddings, Layers, Optimizers, Epochs, Learning Rates to see how it affects the performance of the Model.

Report: I did a lot of parameter tuning, and I have built a table for the performance comparison of all of my “versions” of the Neural Network. Here I am judging the performance of my model based on the MSE value on the test data.   
  
  
I will go into detail about each parameter that I tuned and my observation regarding that.   
  
**1) Number of Layers**: The first layer of my model contains vectors that have the information about the users and the movies, and this is constant across all the variations of the models as this is essential to the functioning of the model. After that, I had either 1 or 2 densely connected layers to improve performance, and that is what the screenshot indicates. 2(50, 50) represents two dense layers having 50 nodes each. What I observed is that more densely connected layers do not necessarily mean better performance. After a point the model can overfit and it does not always make sense to have more layers because it does not guarantee a better result.   
**2) Dropout Values**: Pretty much all the reading I did on sample models and regarding Neural Networks had dropout layers after densely connected layers. It makes sense to have these because otherwise the model almost always overfits, giving a great validation result during the training but a definite drop in performance during testing. Here I do fully understand the problem that was discussed a lot during the course – “How to make Neural Networks better explain themselves.” One version of the model, which retained everything from its previous iteration (see 3rd row in the screenshot) except that the dropout value changed from 0.2 to 0.5. The MSE value went from 0.81 to 1304561. To me this makes no sense, and I can’t explain it, which is why I kind of wish that Neural Nets weren’t such a black box.   
**3) Optimization:** I didn’t change optimizers until pretty late in the tuning phase, because the one I was using by default is Adam, which in my experience will always give good results. So I ran optimizer changes only on the configuration that gave me the best results in the first round of testing (2nd row in the screenshot). As I had expected, Adam beat pretty much all the other optimizers Keras has to offer. I do know the theory and the math behind some of these optimizers (SGD, RMSprop) but not all, and again I couldn’t explain exactly why Adam was performing better compared to the others. It was just a matter of trial and error.   
**4) learning rate:** A lot of the Keras pages and Neural Net explanations had 0.001 as the learning rate so I decided to keep that for most of my testing. I know that a slower learning rate over a larger number of epochs is supposed to be the best combination but sometimes it can tend to overfit. 0.001 is a low enough number to allow the network to fully capture the features of the data without taking a super long time to train.   
**5) Epochs:** My first ever test was on 10 epochs (not included in the chart), and because Keras shows validation loss and validation MSE, I noticed that the best result always was around epoch 4 after which the model started to overfit. I’ve thus kept 4 epochs for most of my tests, and you can see that when I increased the number the MSE number took a hit.

## N1-F: Build a second Model that also takes into account the genre of the movie as a factor in recommending items to a user

Report: I feel like to some degree the implicit nature of the recommendation system would take the genre into account. However, I have not explicitly added this feature and I have effectively dropped this requirement from the final version of the project.

## N2-F: Use a web scraper to attach movie posters to the results while displaying recommendations, to improve the experience.

Report: I had some trouble getting the correct image to load and show properly as a Pandas result. This functionality would be better used if I had built a web-interface or a GUI for my project. But as things stand, the images ruin the result of the pandas dataframe since I have removed the code related to this and this requirement is dropped.

## N3-L: Look at the scalability of the code. The movielens dataset is available in various sizes. Once the project is perfected on a smaller scale on the local machine, try to run it on larger sizes with more computing power on the cloud.

Report: I would consider this requirement as complete, because my code definitely works on the largest version of the dataset that MovieLens has to offer. I have been doing my parameter tuning on my local machine with the smaller dataset, but I plan to take my top 2 configurations and run them on the cloud on the larger dataset.

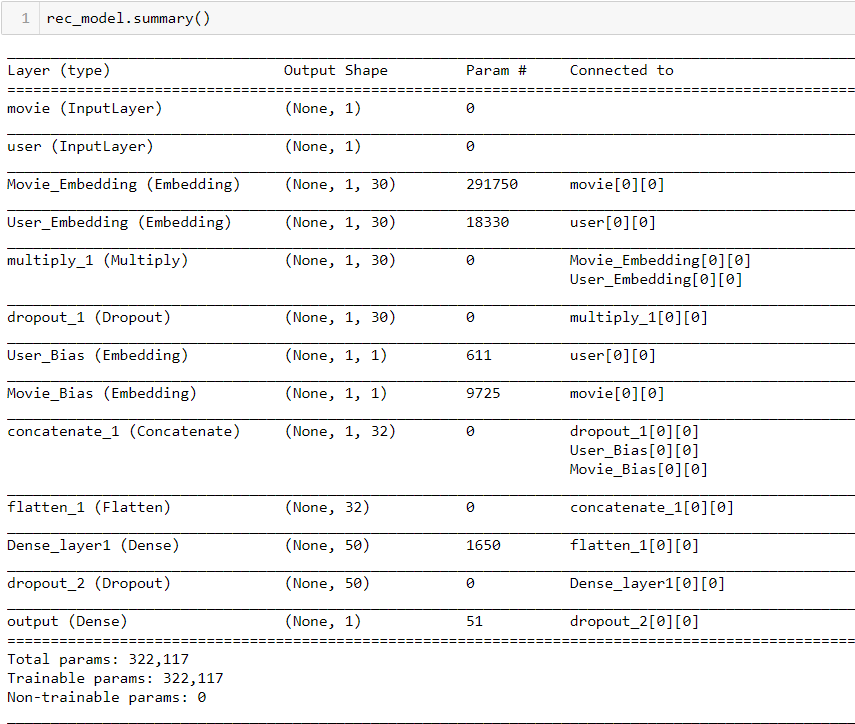
## 6.3 Report on Design v2

Describe the design that you have used so far. Indicate how, where, and why it differed from your planned design. Describe the advantages and shortcomings of your design. Include at least one diagram to refer to in your narrative. Underline edited sentences and additions from v1, if any.

1. Pre-process the data:
   1. This includes looking at the shape of the data, removing inconsistencies and garbage values etc.
2. Split the data into train and test:
   1. This will most likely be a 80-20 or 70-30 split, using a library like sklearn.train\_test\_split[1]
3. Defining the Model
   1. Creating separate embeddings[2] for the Movies matrix and the Users Matrix
   2. Merging the embedded matrices using a dot product
   3. Applying several other Neural Network Layers[3] such a Flattening, Dense Layers, Dropout Layers after the dot product of the matrices. (In a combination that gives the best results, as I will be experimenting with this)
4. Deciding parameters such as Optimizers, Epochs, etc
5. Train the model
6. Write an interface function that interacts with the user to take input such as a userId (to get top recommendations) or a <user, movie> pair (to get the predicted rating)
7. Implement the web scraper in the interface function to show movie posters along with the results.

This is the design I had submitted as a part of Assignment 4, and it remained unchanged over this week.

The internal design of my Neural Network is shown in the screenshot below.



The number of dense layers and the dropout values changed according to the parameter tuning process I explained earlier.

**Why use Embedding Layers?** As you can see in the model summary, we Multiply the user and the movie vectors as a part of Collaborative Filtering. This creates a pretty sparse matrix. Embedding layers help the network deal with sparse matrices better. What an embedding layer basically does is convert the input into lower dimensions that are easier to train and understand for the network. [1]

**Why use Dropout Layers?** Dropout layers are, as I explained earlier, essential to make sure that the model does not overfit. A dropout layer essentially turns off a certain percentage of Neurons during a forward or backward pass; essentially shutting down those parts of the network.   
  
**Why use Dropout Layers?** A dense layer is a fully connected layer where each neuron of the Dense Layer is connected to each neuron of the next layer. A densely connected layer provides learning features from all the combinations of the features of the previous layer, whereas a convolutional layer relies on consistent features with a small repetitive field. [2]

## 6.4 Tools v2

Describe the tool(s) that you are using. Show samples. Describe their advantages and their shortcomings. Limit: 1 page of 12-point Times New Roman. Underline edited sentences and additions from v1, if any.

Collaborative Filtering: This is a technique used in Recommendation Systems. This method focuses on finding “users like you” to make recommendations. The implementation involves creating a product of the user and movie matrices, which usually results in a pretty sparse matrix. I have used Neural Networks along with Collaborative Filtering but it can also be done in a purely mathematical sense without “learning algorithms”

Neural Networks: Since all of my data was numerical and neural nets work really well with numerical data it made sense to go with neural networks. The only disadvantage is that they’re somewhat of a blackbox. When I was tuning parameters there was a lot of trial and error without understanding why a particular change is better or worse. However, the results of the system are great.

## 6.5 Contrast between approaches

You were to include two technologies or approaches to your problem, and implement at least one. Contrast the two technologies (not the tools—those are described above) as they specifically relate to your project.

At the beginning of the course, the two approaches I had considered were Neural Networks and KNN:

I implemented Collaborative filtering in Neural Networks, which is essentially recommendations based on finding users similar to the input user. KNN works in a similar fashion, where we find N nearest neighbors and give results based on the neighbors.

However, a KNN algorithm would not be very good in terms of the time it takes to train. The scalability would be poor as well.

## 6.6 What did *not* work well

We want to see that you understand limitations, not just benefits. Explain the most significant aspects of your project that fell short of your plans or desires.

During the tuning of Hyperparameters, I did not always understand why a certain change in a value like dropout or learning rate affected my result the way it did. Although Neural Networks are a black box which makes it difficult to understand the “why” of some methods, the theory behind the parameters gives some clue as to how the tuning of a parameter should affect the result. However when it doesn’t work out that way, it is difficult to explain why.

## 6.7 What *did* work well

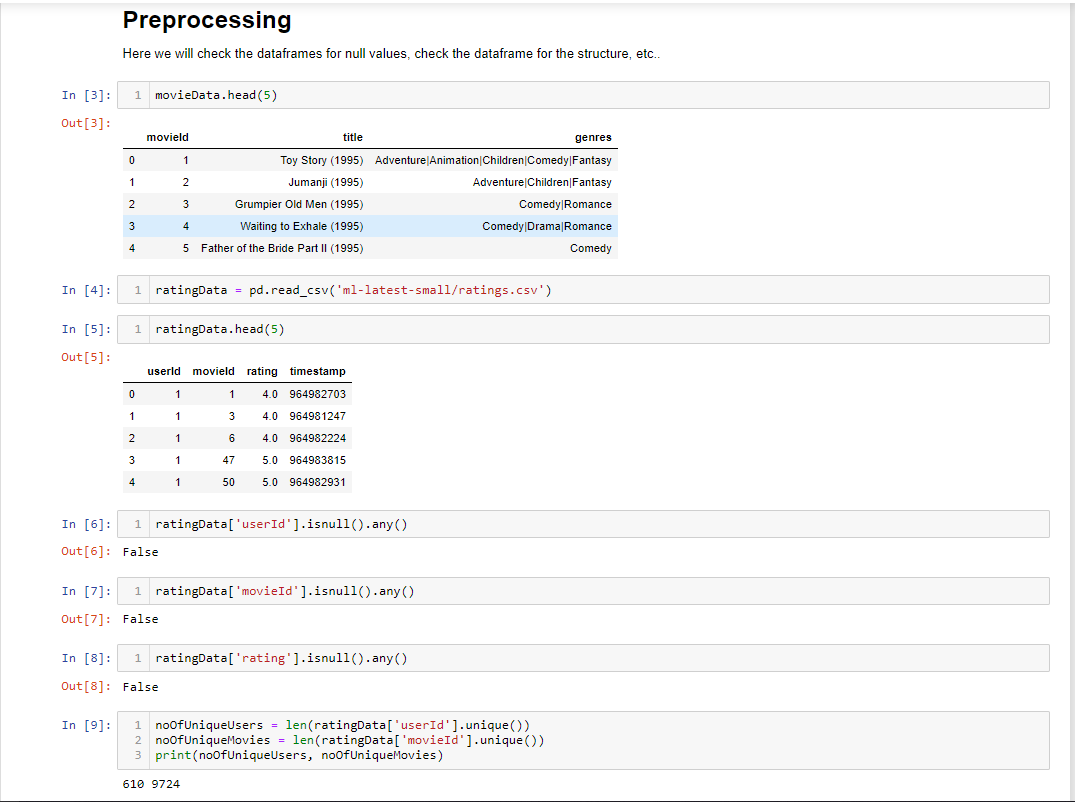
In paragraph form, explain the most significant aspects of your project that met or exceeded your plans or desires.

The model is highly robust and scalable, which are typically the features of Neural Networks. Once the model is trained it is easy to quickly extract results from the models. The Neural Network works extremely well and has a very low error.

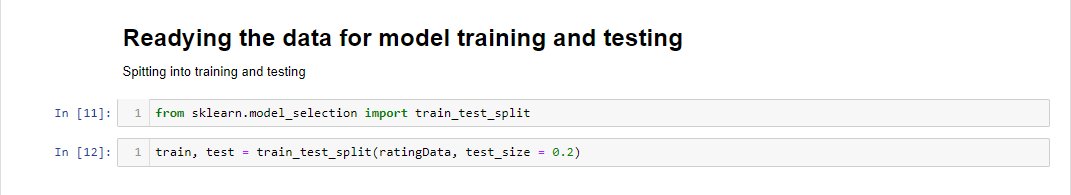
## 6.8 Sample Source

Supply key excerpts from your source code—or what comes closest to “source code.” Limit: 2 pages of 12-point Times New Roman equivalent. Include an explanation of where the excerpts fit in your implementation. These are counted as figures, and do not count towards the total page limit.

### Pre-Processing the Data:

Since the data came from a really good source, I did not have to transform or filter any of it because the data was good.   
  
  
I check for any null values, and look at the shape of the data.

### Splitting the data into Training and Testing sets:

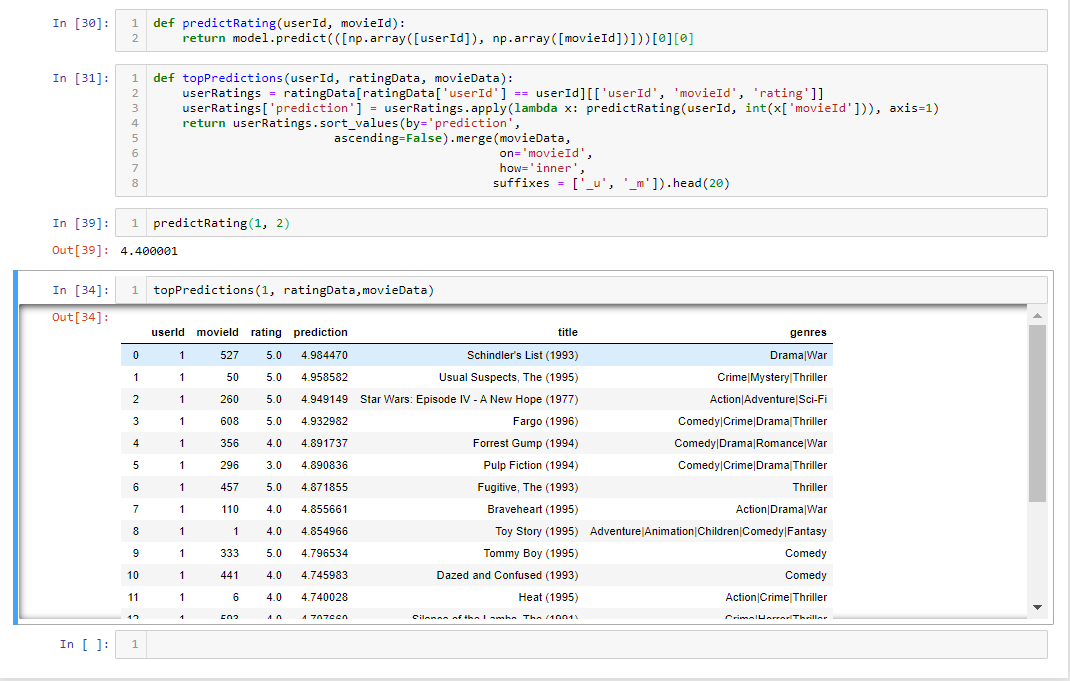
  
  
I have used the sklearn library ‘train\_test\_split’ to separate my data into training and testing sets. I have kept 80% of my data for training and 20% for testing.

### Coding the Model:



- I first embed the arrays containing the movieIds and userIds.  
- The embedding layers are then multiplied to form the utility matrix, which is the cornerstone for the Collaborative Filtering method.   
- the input layers are then concatenated and ready to be fed into the actual neural network  
- the model consists of 1 (or 2) dense layers depending on the configuration.

### The interface functions:



The function predictRating returns the predicted rating for a <user, movie> pair.  
The function topPredictions gives the top 10 movies with the highest rating that the user has not yet seen.

## 6.9 Source

Refer the reader to your source code (or what comes closest to it) and input where possible.

Attaching the Jupyter Notebook file along with the submission.

## 6.10 Presentation

Make a 3 minute video (5 minute maximum if you have to) presentation of your results, including a demonstration, and point your facilitator to the location of the video.

Link to the video : <https://drive.google.com/file/d/1dasmzyUprQ6o1IPTBe1AZIffgnN2_QFG/view?usp=sharing>

# Evaluation

