Problem 1: One-Vector-Per-Item Collaborative Filtering with SGD and Autograd

a)

RMSE vs. Epoch for various values of K:

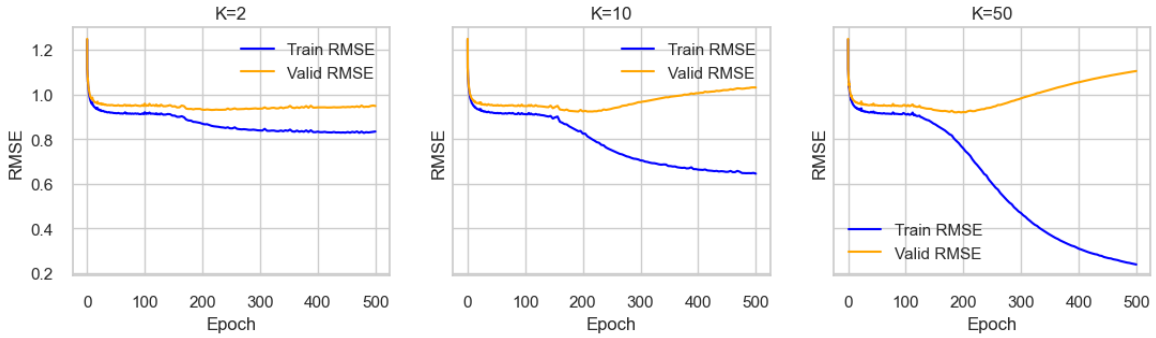


Figure 1: **(i)** As K increases, we see that the model has a greater tendency to overfit over many epochs. With more factors, the model can fit more closely to any noise in the training data, which explains this behavior. **(ii)** The 'best' validation set performance improved with more factors, going from 0.9301 for $K = 2$ to 0.9191 for $K = 50$. This suggests that more factors allow us to better model the relationship between users and movies, as long as we stop early. **(iii)** We chose a step size of 0.8 because it showed a steady training curve with clear overfitting, unlike larger steps of 1, which shows divergence, or smaller values of 0.05, which did not show interesting results within 500 epochs.

b)

RMSE vs. Epoch for $K = 50$ and $\alpha = 0.1$:

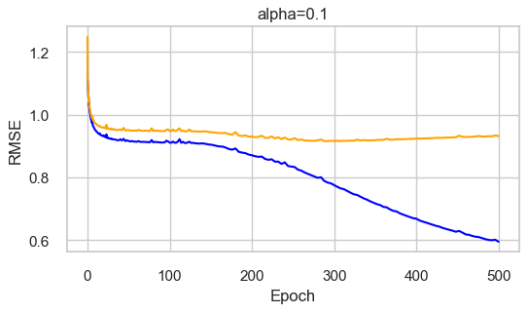


Figure 2: **(i)** We selected $\alpha = 0.1$ because we found that it allowed us to achieve the lowest validation error (with early stopping) across many candidate values (0.01, 0.05, 0.08, 0.1, 0.2, 0.5, 1, 10). This was a good tradeoff between fitting the data while limiting the model complexity to avoid overfitting so our model can generalize well. **(ii)** We stuck with a step size of 0.8 because it continued to provide a steady training curve where we could see the validation error minimum and overfitting within a reasonable number of epochs without diverging. **(iii)** This regularization improved the minimum RMSE of our validation set, which was 0.9191 for $\alpha = 0$ and 0.9161 for $\alpha = 0.1$. The simpler model encouraged by the penalty term performs *slightly* better on unseen data.

c)

Train/Val/Test performance for each model:

Model	Train RMSE	Valid RMSE	Test RMSE	Train MAE	Valid MAE	Test MAE
$K = 2$	0.859	0.930	0.931	0.676	0.732	0.727
$K = 10$	0.835	0.923	0.921	0.657	0.728	0.721
$K = 50$, $\alpha = 0$	0.769	0.919	0.914	0.604	0.725	0.715
$K = 50$, $\alpha = 0.1$	0.595	0.933	0.925	0.468	0.737	0.725

Table 1: Based on RMSE, we would recommend using 50 factors ($K = 50$), as this model performs the best on both the validation and testing sets in terms of minimum error. This recommendation does not change when using MAE, since the $K = 50$ model still performs the best on both sets.

d)

Latent Factor Plot, Selected Movies:

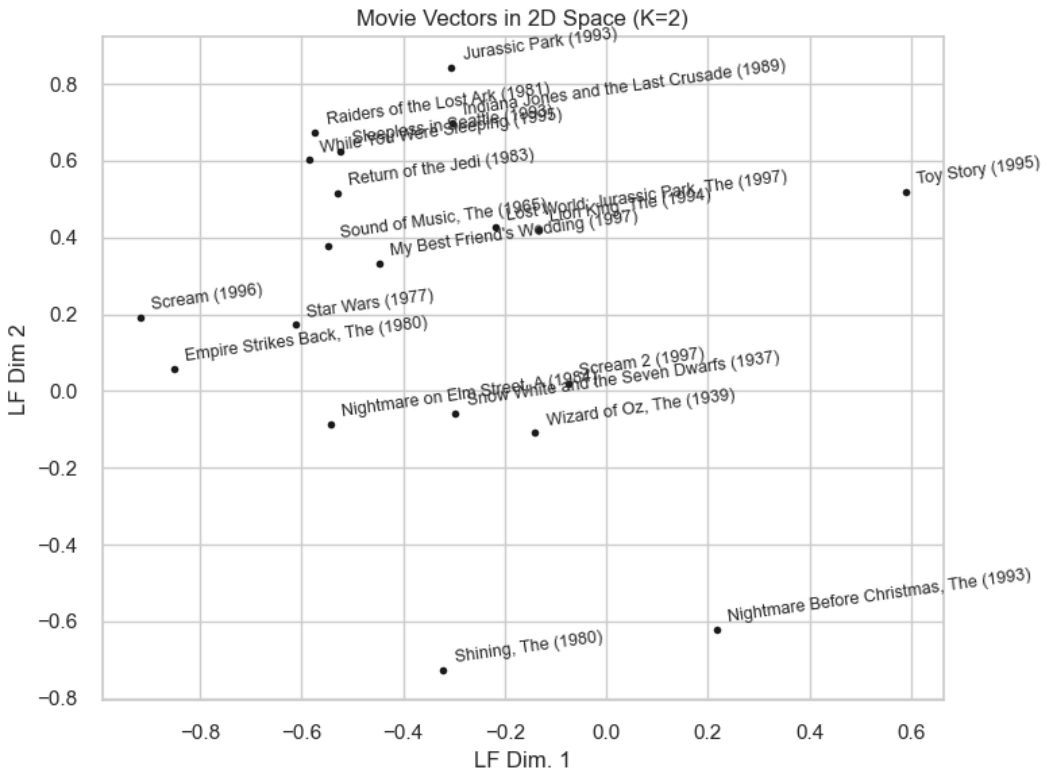


Figure 1: It appears that movies toward the right side of this plot with higher values for Dimension 1 (such as The Wizard of Oz, Toy Story, The Nightmare Before Christmas, and The Lion King) are better suited for a younger audience, while those on the left (Scream, A Nightmare on Elm Street, or Star Wars) might be better suited for an adult audience. In most cases, we also see that movies from the same saga are loosely grouped together, as with Star Wars, Indiana Jones, and Jurrasic Park. That being said, Scream and Scream 2 are strangely far apart on this plot. Additionally, although we can see some patterns in placement on dimensions, there are also some separation between some movies that might be worth noting. Toy Story, The Nightmare Before Christmas, and The Shining all seem to be separated from the main cluster of movies (Toy Story on Dim. 1, and the Nightmare Before Christmas and the Shining on Dim. 2). There is no discernible difference these movies have from the rest, so perhaps there are some thematic/popularity differences between these movies and main cluster we see in the top left.

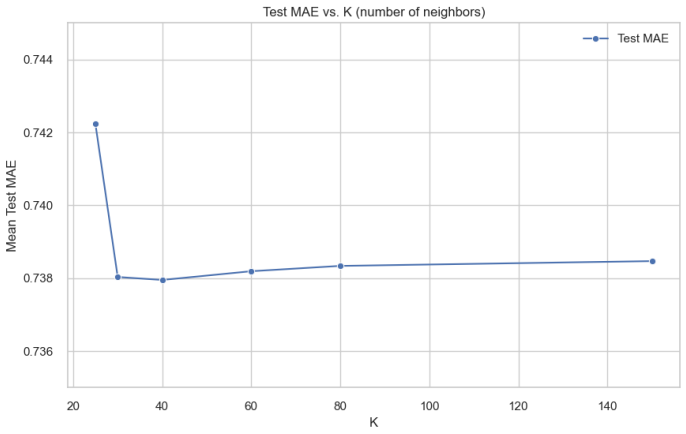
Problem 2: Open-Ended Recommendation Challenge

a)

We decided to use an ensemble of methods - KNN and SVD. We trained and fitted a KNN, SVD, and SVDpp model separately, and then averaged these results in order to take advantage of each of these models and to avoid overfitting from any one of these models in particular. We chose KNN in order to examine similarity in user ratings, and SVD to leverage latent factors in the data through matrix factorization. Additionally, SVDpp takes into account implicit ratings, which while more computationally expensive, may provide important aspects about the data. For KNN and SVD, we used GridSearchCV in order to choose our hyperparameters (focusing on k values for KNN and number of factors and epochs, regularization, and learning rate for SVD). We used RandomizedSearchCV for SVDpp since it was too computationally expensive to do GridSearchCV in a reasonable time. For SVDpp, we optimized the same parameters as we did for SVD. We fit the Grid/Randomized Search objects on the training data and then found the model that yielded the lowest MAE. We then fit the best model on the training data, and then made predictions on the test data.

b)

Selection of K (# neighbors):



This plot shows the search process for our hyperparameter K in the Suprise KNNBaseline model. This hyperparameter controls the number of neighbors used to predict the rating of a new user/item pair. When this value is small, like $K = 25$ we see noticeably higher MAE, likely due to overfitting to the noise of just a few training examples. We can expect this low K to provide a complex decision boundary around the training examples that does not generalize well to unseen data. On the other

hand, very large K values could underfit to relationships between users and movies, as the model averages predictions over a larger neighborhood and potentially misses important patterns. We can expect smoother decision boundaries here, with a lack of flexibility to individual examples. Higher values of K are also more computationally expensive, as the prediction step has to account for more neighbors. Taking these factors into account, we decided to use $K = 40$, which matches the value of K selected by our grid search.

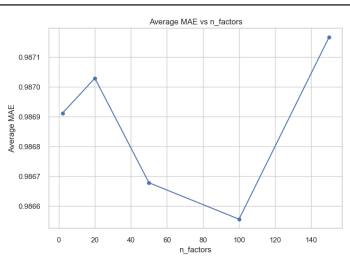
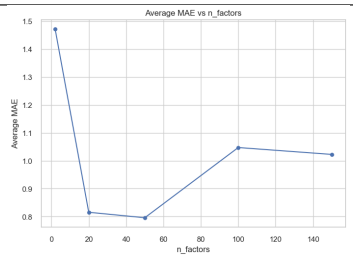
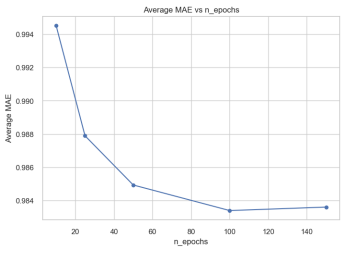
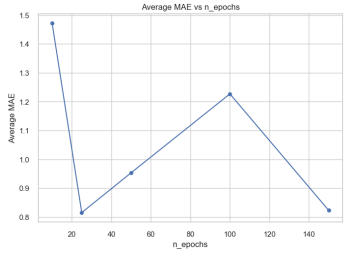
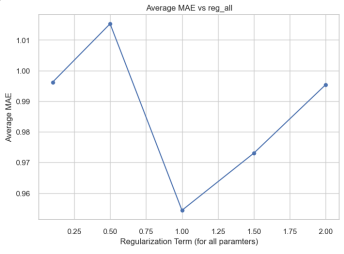
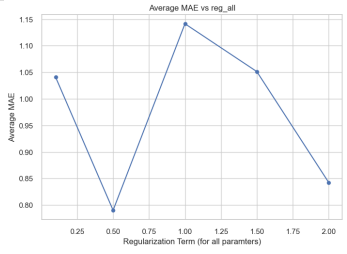
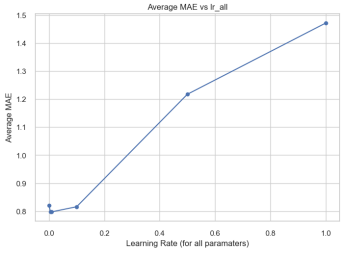
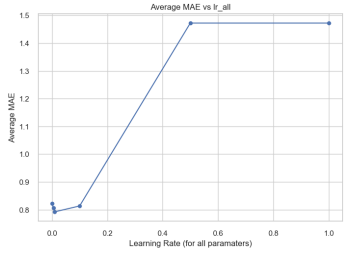
	SVD	SVDpp
# factors		
# epochs		
Regularization		
Learning Rate		

Figure 4(cont.): Above are plots illustrating the average MAE of held out data given $n\_factors$, n_epochs, reg_all, and lr_all for both our SVD and SVDpp model. An important distinction to note is that we used GridSearchCV for SVD and RandomizedSearchCV for SVDpp for computational cost reasons, so not all combinations of hyperparameters were tested in SVDpp. In these graphs, we see evidence of overfitting. For example, the plots examining $n\_factors$ both illustrate a decrease in average MAE until a certain point, after which it increases. With SVD, it is 100, while with SVDpp, it is around 50. This makes sense since SVDpp is a more complex model than SVD, and increasing the complexity would cause overfitting earlier in SVDpp than in SVD. We see a similar phenomenon with n_epochs. With SVD, average MAE decreases and then plateaus/slightly increases after 100 epochs. With SVDpp, average MAE is more variable, perhaps due to the random sampling of hyperparamters in RandomizedSearchCV. We see optimal regularization valued at 1 and .5 for SVD and SVDpp, respectively. This value should ideally help with overfitting since it penalizes overly complex models We also see optimal learning rates (step size) at around .005 for both SVD and SVDpp.

c)

	MAE on test split	MAE on leaderboard
Problem 2	0.738	0.718
Problem 1c	0.715	N/A
KNN (alone)	0.807	N/A
SVD (alone)	0.726	N/A
SVDpp (alone)	0.728	N/A

i. Our leaderboard score was lower than the test split of the dev data. This means that our model performed better than expected on new, unseen data, meaning it generalized well. This also makes sense because we trained our model on the full training set without withholding data when submitting to the leaderboard.

ii. The MAE of the Problem 2 model on the leaderboard was similar to the Problem 1c performance on the test splitting, meaning they performed similarly on never-before-seen data. It was somewhat surprising to see that our hybrid model did not significantly outperform the one-vector-per-item collaborative filtering model from part 1, given that our final model from part 2 considered varying predictions from different models to boost its performance. This is a testament to the fact that sometimes a simpler model is just as suited for the task at hand as a complex model.

d) Discussion

One pro of this approach was we were able to leverage the results of different models we thought could make accurate predictions of user ratings. We sought to reduce the overfitting of one particular model by averaging their results using an ensemble of models. With KNN, we were able to examine similarity in user ratings, and with SVD and SVDpp we were able to leverage latent factors and implicit ratings, respectively. As one can expect, this method of using different models is computationally expensive, as we have to use a hyperparamter search on 3 seperate models and train them separately. Additionally, by using an ensemble of methods, it is harder to pinpoint how the model is coming to certain conclusions since multiple models are contributing to the end result. Any problem where computational time/expense is of concern is not well suited for this ensemble of models. However, problems prioritizing accuracy over computational cost may find this method useful in order to leverage different aspects of different models.

If we had another week, a model we were curious about was LightFM. We thought this model would effectively combine user data (gender, age, etc.) with user rating data, as perhaps some of this data correlates with certain patterns in the data that our model could leverage. We had some issues with data processing with LightFM, so we decided to focus on Surprise's SVD and SVDpp, but would probably spend more time trying to utilize user data to make more predicitions if given the opportunity.

One key takeaway from this project is the importance of hyperparameter adjustments, as they can greatly impact the MAE. Additionally, balancing computational complexity, since hyperparamter fitting for SVDpp in particular was unreasonably expensive and time consuming. We also learned about more about implementing and training collaborative filtering methods with problem 1.