|  |  |  |
| --- | --- | --- |
| Slide 1 |  |  |
| Slide 2 |  |  |
| Slide 3 |  |  |
| Slide 4 |  |  |
| Slide 5 |  | The bar chart displays count vs. genre from the course dataframe.  We can see that backend dev genre is the most occurrence within the dataset whereas chatbox and blockchain have the least occurrences.  We can also see that the count ranges from 4 – 78 with the total sum over 307 (number of courses within the dataset) as expected since there will be some overlap of genre per courses. |
| Slide 6 |  | This is a histogram of user count vs. course enrolls to observe the distribution.  We can see that majority of users only enroll to a single course.  however, if they do enroll for more than one, they do seem to continue enrolling to more courses with the curve approaching a plateau toward ~50 enrolls. |
| Slide 7 |  | Here we have top 20 most popular courses from the dataset.  We can see that python for data science is the most popular course with 14936 enrolls followed by introduction to data science with 14477 and big data 101 13291.  The 20th popular course is data privacy fundamentals with the enrollment of 3624 |
| Slide 8 |  | The wordcloud of the course titles show the popularity of key words in the course titles such as data science, python, machine learning, and big data. |
| Slide 9 |  |  |
| Slide 10 |  | The categorical features are extracted from raw data in form of profile and course genres.  Both profile and course genre is then turned into vectors for score calculation.  The recommendation score is then calculated via dot product of a course vector and a profile vector.  The only hyperparameter in recommender system is score threshold. If the score is higher than the threshold, the course will be recommended to the user.  The threshold value is then fine tuned to adjust the number of recommendations. |
| Slide 11 |  | Iterating through threshold score of 1 – 39  The number of recommendation goes down over increasing threshold as expected.  The top 10 was found from the recommendations using score threshold of 10 |
| Slide 12 |  | The raw data consist of courses with their title and description.  The feature engineering step takes the course titles and create bag of words features per course.  The pairwise similarity score was calculated based on bow features between all pair of courses resulting in the similarity score matrix.  The recommendations were then generated by using each user’s enrolled course information as an input. Here, we are excluding the enrolled course from the list of available courses to avoid recommending the same course.  The similarity score is retrieved from the score matrix calculated in step 3 between the enrolled course and the rest of available courses. Each recommendation is required to pass a certain similarity score threshold which can be fine tuned to control the size and quality of recommendations. |
| Slide 13 |  | The graph displays average number of recommendations per similarity score ranging from 0.4 ~ 0.8  At 0.7 similarity score, the average number of recommendation approaches below 1 indicating that most courses have less than 0.7 similarity score given the dataset |
| Slide 14 |  | Data processing step takes care of unbalanced dataset and uses standardscaler to normalized the skew  Feature engineering step includes PCA and clustering (ie: k-means) to reduce the dimensionality of the dataset as well as adding on cluster label as a new feature.  The features are then re-grouped based on the cluster label and user.  For each input (user), the matching cluster is searched for the list of available courses, which is then evaluated based on the threshold popularity of the course within the cluster.  If it passes the evaluation, the courses within the cluster will be recommended. |
| Slide 15 |  | The average recommendation vs. popularity threshold graph shows that at threshold beyond ~75, the number of recommendation approaches near 0 |
| Slide 16 |  |  |
| Slide 17 |  | The raw data is cleaned up and engineered to retrieve relevant features.  The dataset of features are then split into train and test set  KNN model trained based on train set.  The model is then tested on the test set for validation.  The hyperparameters such as min/max number of neighbours as well as distance metrics can be further tuned until the desired validation result is achieved.  The model can be used to predict score/rating of an unknown items based on their similarity, which is then used to provide recommendation (ie: via collaborative filtering) |
| Slide 18 |  | After clean up and engineering features, NMF will be applied to the dataset (m \* n) to generate factorized matrices of non-negative features.  NMF can be used to estimate the original rating which can be then used to generate the estimated rating while reducing its dimensionality as well as highlighting the important features.  The estimated ratings then can be used to generate recommendations by collaborative filter. |
| Slide 19 |  | The features are processed into one hot encoded categorical features per feature and fed into the embedding layer of neural network.  The neural network will construct a latent features from the input which will be then evaluated to predict the outcome.  The neural network will be trained to minimize the loss function on train set.  Once the model is trained, the further optimization can be done by tuning the hyperparameters.  The model then can be used to predict the outcome (ie: probability of completing the course) based on unknown input (ie: user, course)  Based on the evaluation (ie: probability threshold), the recommendation then can be generated. |
| Slide 20 |  | The graph displays RMSE value of train and validation set over 10 epochs  The RMSE value seem slowly decrease with the epochs though not as much as the train value |
| Slide 21 |  | 7 models were built for RMSE comparison  SVD: n\_factors = 100, epochs =20, yields RMSE value of 0.2627  Random: yields RMSE value of 0.2615  NMF: n\_factors = 15, epochs = 50 yields RMSE value of 0.2040  KNN: k = 40, yields RMSE value of 0.1958  Baseline: yields RMSE value of 0.1598  KNNBaseline: k=40, yields RMSE value of 0.1440  Embedding: 16 embedding layers, yields RMSE value of 0.1190  The result shows that SVD performed the worst with RMSE value of 0.2647 and embedding with the best RMSE value of 0.1190 |
| Slide 22 |  |  |
| Slide 23 |  |  |