University of Ottawa

Faculty of Electrical and Computer Engineering

uOttawa

GNG 5125/Data Science Applications

Professor: Arya Rahgozar

Project Report on Tourist Recommender System

Submitted by

Yusri Al-Sanaani	300216450
Hetvi Soni	300200976
Tavleen Kour	300213090
Immanuella Iyawe	300150838

Monday, April 12, 2021

Table of Contents

1. Introduction

It is often essential for travelers to consult different people while making an itinerary for a holiday and as such, exchanging such information over the internet and social networks have become immensely popular. There is a vast amount of information available online but at times it becomes a little difficult to obtain the right type of information as per a user's choice.

This is the place where travel recommendation systems come into the picture and they have become quite popular in the last few years. Travel recommendation systems can be based on various features like location type, the best time to visit the location, experiences of past visits different users, cost, visiting hours, etc.

Recommendation systems are basically of two types: content-based and collaborative filtering. To get a better understanding here we will discuss the types of recommendation along with their advantages and disadvantages.

Content-based recommender systems try to recommend items similar to those a given user has liked in the past. ("Recommender Systems - Brigham Young University") The basic idea of the content-based systems is matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of a content object (item), to recommend new interesting items to the user.

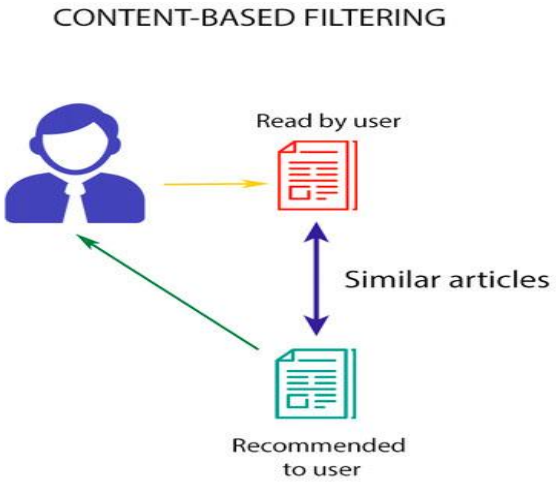


Figure 1: Content-based recommender systems.

Advantages:

- User dependence
- Transparency
- New item

Disadvantage:

- Limited content analysis
- New user
- Overspecialization

Collaborative filtering based models use the collaborative power of the ratings provided by multiple users to make recommendations. ("New contextual collaborative filtering system with ...") The basic

idea of collaborative filtering methods is that the unspecified ratings can be inputted because the observed ratings are often highly correlated across various users and items.

Figure 2: Collaborative filtering-based recommender systems.

Advantage:

- Good serendipity
- No domain knowledge required

Disadvantage:

- Generates a sparse matrix
- Recommendation based on neighboring users.

2. Problem formulation

Implement a recommendation system based on content and collaborative filtering, leveraging many algorithms to provide different prediction features such as providing recommendations based on:

- Attraction type
- Targeted province
- The best time to visit the attraction
- Multiple combined features

Choosing the best performing model and implemented a chatbot for it. Therefore, the recommender system turned out to be very useful and convenient to use for the tourists. The proposed goal of this system is to identify the attractions the tourists want to visit in Canada as demonstrated by their post-viewing rating (user-satisfaction metric) and identify a list of attractions recommendations, which contains at least one that the users will visit as their next selection (engagement metric), resulting with multi-functional recommender system that based on content, context, and ratings.

2.1. Motivation to choose this topic

The problem of endlessly sieving through so many web pages/searches isn't the only hassle travelers have to face while seeking the best destinations or their choice of destinations. More work is also put into travel preparations that is, getting information specific to the destination of choice and what the

requirements as per what to expect on the journey can also be cumbersome hence the need for a travel recommendation system that is content-aware.

2.2. Objectives:

For a travel attractions recommendation system, the primary objective is to suggest tourist attractions to the user but many other factors are essential to make the recommender system a successful model.
Here, our main objective is to address the following factors and come up with a unique model.

- **Relevance**:

Relevance is the primary objective that we are trying to achieve in our system as it is essential to predict certain travel attractions that the user might be interested in. It is very important to predict a location that the user might be interested in based on his previous choices.

- **Novelty:**

Here, to bring novelty what we try to do is that we try to predict/suggest a location that the user has not been to before.

- **Serendipity**:

Here what we try to do is predicting/suggesting something completely different to the user based on his previous choices for travel attraction.

- **Increase the diversity of recommendation output:**

Here we are trying to enhance our model and by providing good top-n travel recommendations to the user. so that the user does not get bored with the same type of recommendations.

Claims:

- We aim to develop a system based on a content/context-aware model to suggest new travel locations to the user based on his previous choices.
- We also aim to suggest locations based on the location type user wants to visit.
- We have implemented a classification model using both sci-kit learn and surprise model and we would compare accuracy and predictions for both the libraries and choose the best one for recommender systems.
- We also aim to produce results using clustering model and study the accuracy for the same.

2.3. Implemented model overview

As shown above, our proposed system begins by scraping the data from the trip advisor website.+ After scraping the data, we performed an Exploratory Data Analysis to get a better understanding of the data. We have designed our recommendation system using both content based and collaborative filtering model.

In the content-based model, the recommendations have been generated using cosine similarity and clustering algorithms. For clustering, we have used K Means, while for collaborative filtering we have implemented classification algorithms using both surprise and sci-kit learn libraries. The algorithms implemented for both libraries are Support Vector Machines (SVM), K-Nearest Neighbors, and Decision Tree. Finally, the results from all the models are analyzed and the best model has been integrated with dialog flow to create a chatbot that makes our system user-friendly and easy to use.

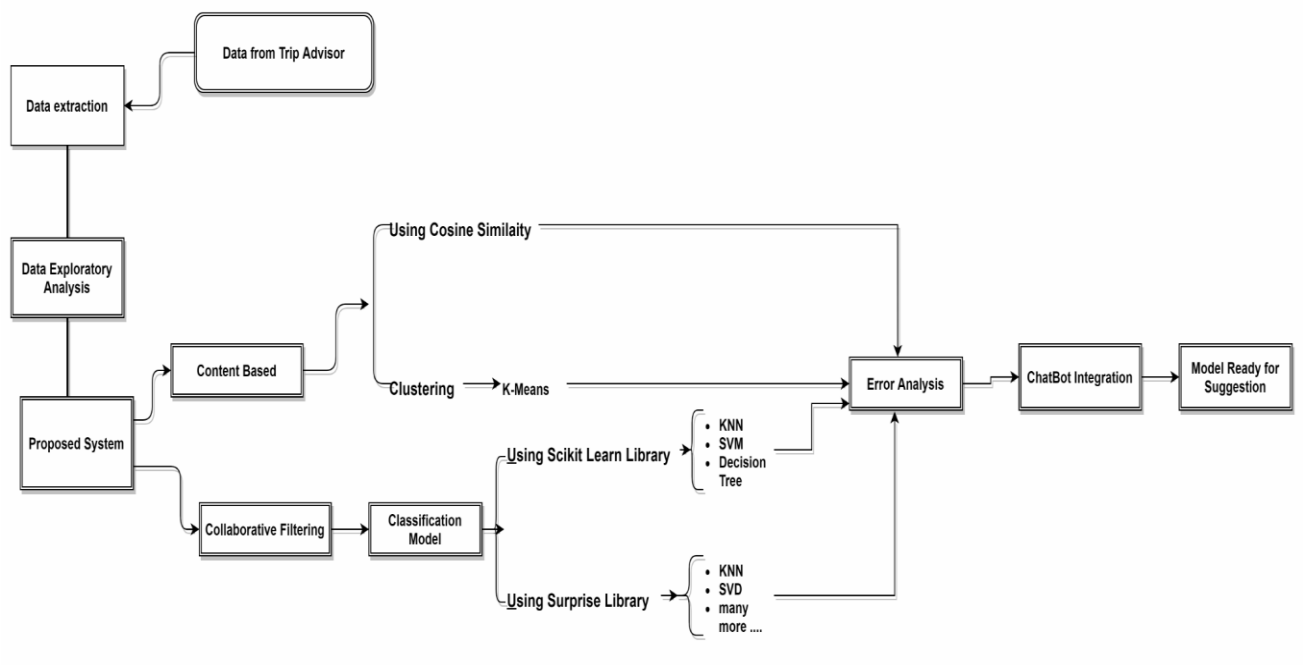


Figure 3: Implementation of proposed recommender systems.

2.4. Libraries used

1. For data extraction and integration.
   Beautiful Soup URL open Requests Pandas Regular Expression
2. Data exploratory analysis.
   Plotly Pandas Numpy Jupyter Dash
3. Content-based recommender system.
   Sklearn Numpy pandas Seaborn matplot
4. Classification-based recommender system.
   Pandas Scikit learn surprise Numpy collection datetime
5. Clustering-based recommender system
   Scikit learn Numpy Pandas Pickle Matplotlib
6. Chatbot integration (dialogflow, flask, and ngrok).

3. Data integration and data preparation

To begin our recommender system, a data set that contained all the required information about various tourist attractions was our main concern. For our recommendation model, we decided to focus on the tourist attractions within Canada and have thus collected various information of several locations. A list of the relevant information used in the system can be shown below.

- Location type (park, lake, historic building etc)
- Location province

- Address
- Best time to visit
- Are there any entry fees?
- User ratings
- User reviews
- Visiting hours

The decision to scrap the relevant information was because no dataset specific to the information needed for the system was readily available online Hence, why we decided to scrape the information from travel websites like TRIP ADVISOR.

All the information in our data set has been collected from a trip advisor webpage that has top tourist sites to visit in Canada. The link we scraped this information is given here at https://www.tripadvisor.in/Attractions-g153339-Activities-Canada.html.

To scrape the information python libraries like beautiful soup and requests were used to collect information from the web pages. The information that we obtained using these libraries are in the HTML text format so get relevant data for the dataset we have used inbuild .get() and .get_text() function to obtain and clean relevant data. Further, cleaning the dataset by removing punctuations, unwanted text and numbers, we have used the regular expression library.

4. Exploratory Data Analysis

Performing the EDA, we decided to visualize the data to have a better understanding of the dataset. The analysis and visualization on the dataset are done using the Plotly library because we wanted to make our analysis graphs to be interactive. Additionally, the data analysis has been divided into various sections to understand the dataset well.

Visualization for locations based on the province:

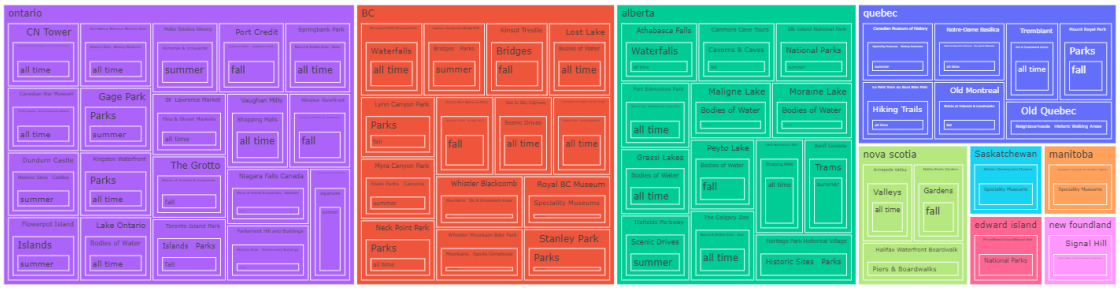


Figure 4: Visualization for locations based on the province (a zoomed image of the tree map).

In **fig 6** above, we included all the tourist locations in Ontario with the location name and best time to visit that place. Here we have also included a zoomed-in image of the treemap produced earlier.

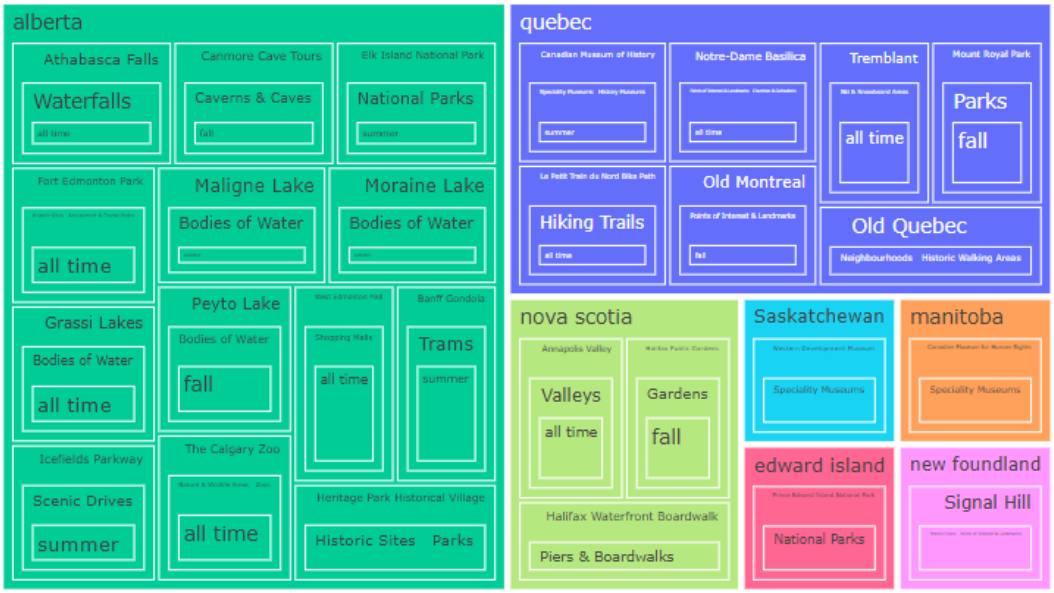


Figure 5: Shows how we have included locations distributed across different provinces and also included information about the type of location, the best time to visit that given location.

Visualization for all the tourist locations in Ontario province.

Figure 6: Visualization for all the tourist locations in Ontario province.

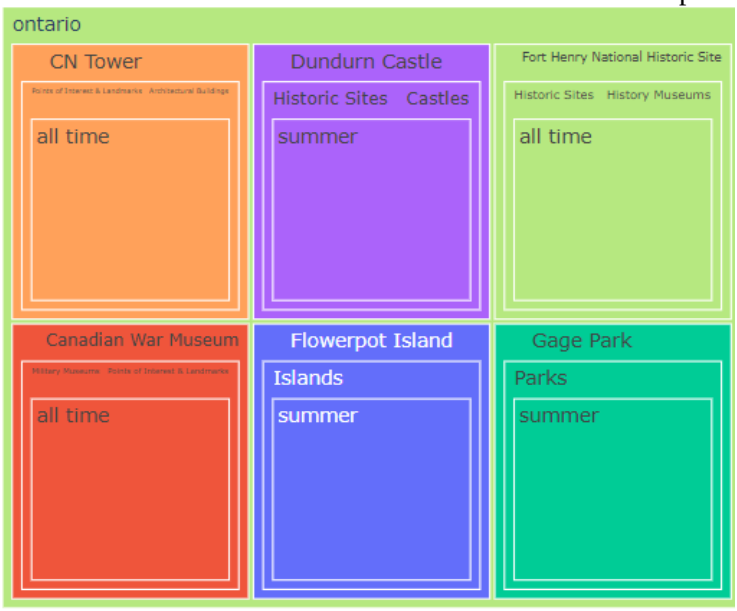


Figure 7: shows a tree map for the Ontario province and we have done a similar analysis for all the provinces.

Best time to visit places in summer along with its ratings.

In this case, we have grouped all the locations based on the best time to visit a location in summer across all the provinces. So one can view the rating for each location by clicking on the location and clicking over each section in the pie chart would generate the ratings for each location.

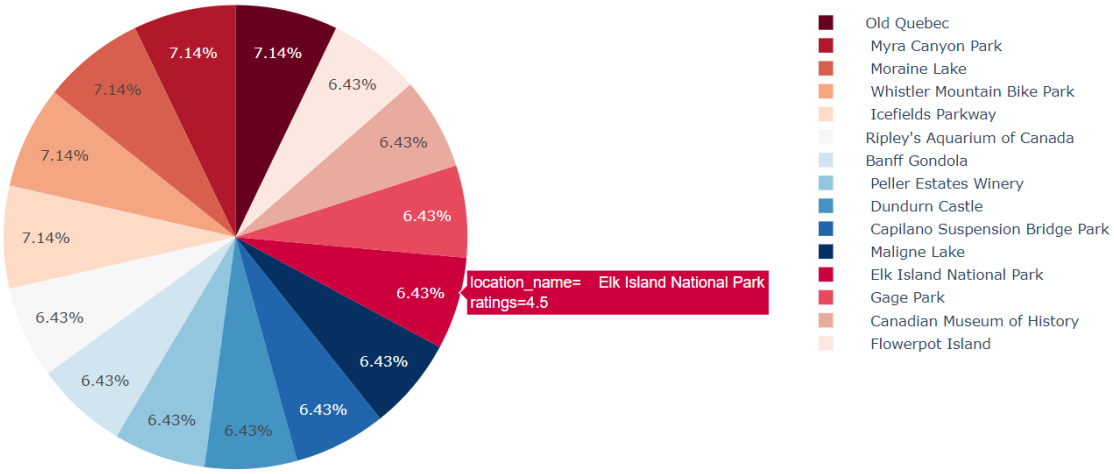


Figure 8: Displays the rating for each location on mouse click.

F**ig 5** displays the rating for each location on mouse click . A similar analysis is also done for fall and summer seasons which can be seen in the collaboratory code.

Locations free to visit.

In this part of visualization, we have grouped all the locations that are free to visit without any entry fee. **Fig 9** below shows the results for grouping the data based on visiting hours, the best time to visit, location name and its province.

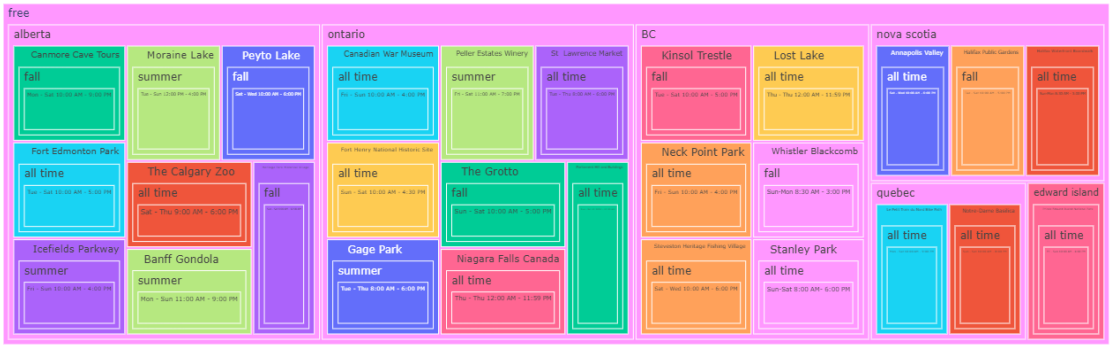


Figure 9: The results for grouping the data based on visiting hours, the best time to visit, location name and its province.

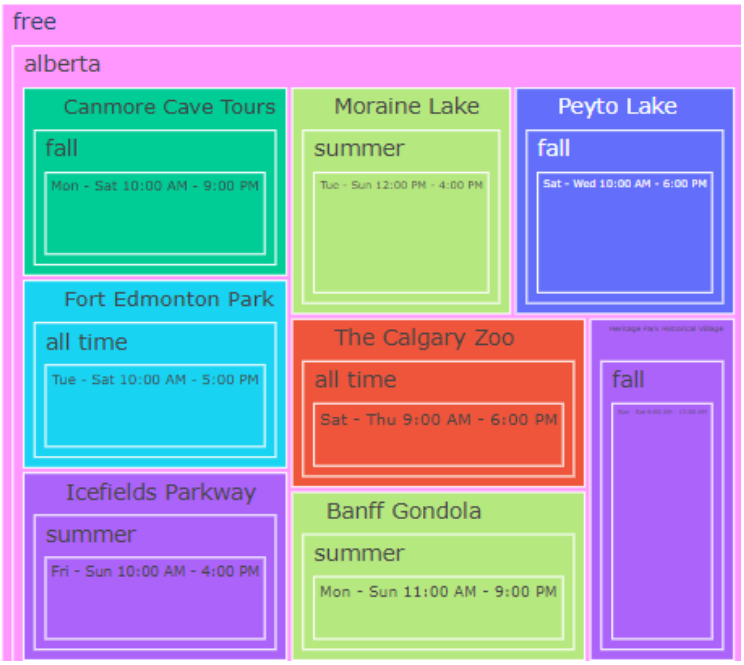


Figure 10: The results for grouping the data based on visiting hours, the best time to visit, location name and its province.

Visualization for tourist locations having more than 10k user reviews.

In this section, we have grouped all the locations with more than 10k reviews across all the provinces.

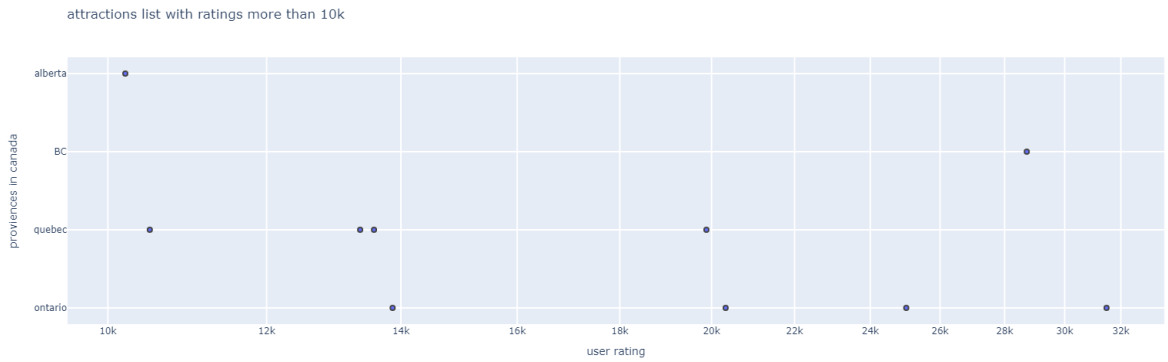


Figure 11: Visualization for tourist locations having more than 10k user reviews.

So to get more information on location and its corresponding location, **fig 11** shows that one can hover over the dots in the scatter plot and all the information would be provided.

Visualization of all locations based on user ratings.

In **fig 12** below we have grouped all the tourist locations based on several user reviews. When one hovers over scatter plot dots one can see the information like location name, its province information.

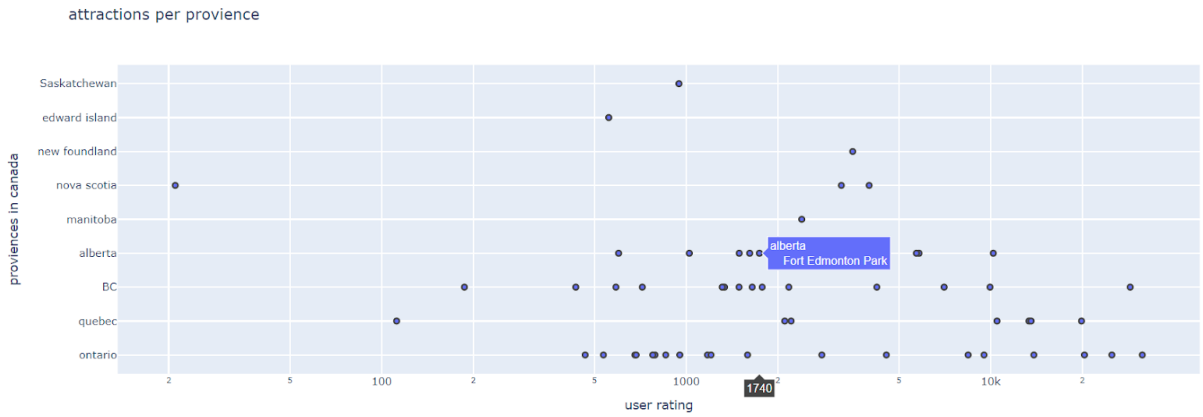


Figure 12: Visualization of all locations based on user ratings.

5. Proposed system

Our proposed system is based on both content-based and collaborative filtering models. In the content-based model, we have implemented the recommender system using two different methods, we have also designed the recommender system based on cosine similarity distance parameters along with that we have used k-means as the clustering algorithm.

So for both the algorithms used in the content-based model, various types of features were used to predict/recommend new locations to the user. The features that we use for the cosine similarity-based model are province, best season, cost, and location type.

Also, for the clustering model, we combined the features like attraction name, province and location type into a single column and then passed it to the k-means clustering algorithm to group them into clusters.

The entire procedure and analytical results for both models are discussed in the next sections. Along with the content-based model, we have implemented the collaborative model using various classification algorithms from surprise and scikit learn library. From the scikit learn library we used KNN, SVM, and Decision Tree, while from the surprise library several algorithms were used like KNN, SVD, SlopeOne, NMF, CoClustering, etc. The accuracy scores for the multiple algorithms were calculated and after a thorough comparison and analysis, the best model for the classification model was selected. The model creation and accuracy will be discussed in detail in the next section.

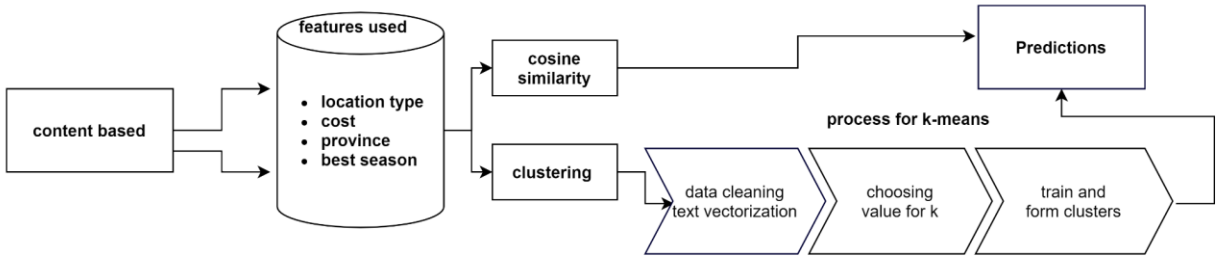


Figure 13: The proposed content based reommender systm.

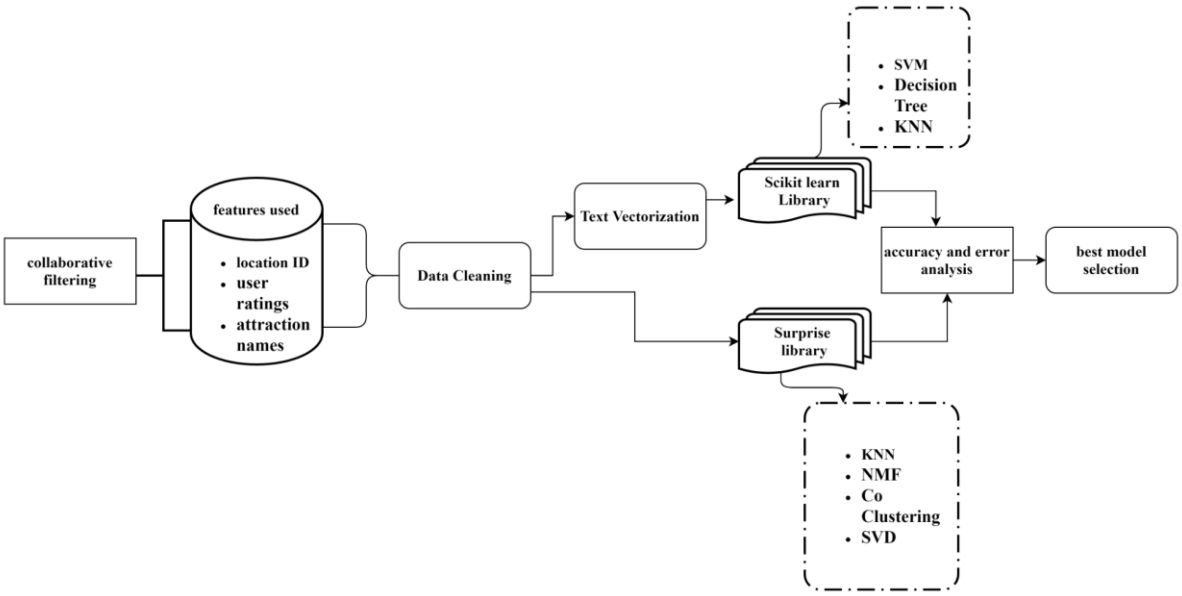


Figure 14: The proposed collaborative based reommender systm.

5.1. Cosine Similarity based Model

The Content-Based Recommender works based on the data that we take from the user, by using this data we create a user profile. This can then used to make suggestions to the user, as the user provides more input or takes more actions on the recommendation thus making the engine more accurate. The block diagram for the entire flow of the model is described in the block diagram below.

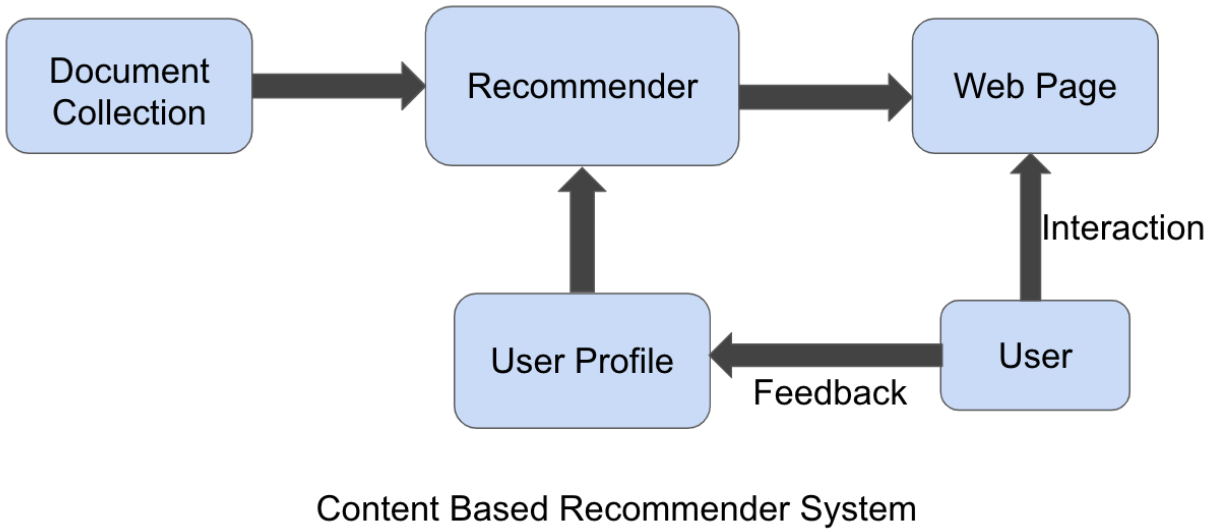


Figure 15: The proposed collaborative based reommender systm.

Modelling the content based recommender system, we first vectorized the data using TF-iDF transformation, the vectorization is performed on three different features.These feature takes inputs from the user, like location type, location province or best season to visit. Based on these inputs the vectorized features are used to calculate the cosine similarity between the input and the vectorized data. After

calculating the cosine similarity the most similar tourist location has been recommended to the user. Code snippet for TF-iDF transformation for location type is as follows

```
]  tfidf_attraction_type = TfidfVectorizer()
   tfidf_attraction_type_matrix = tfidf_attraction_type.fit_transform(df_attractions['location_type'])
   cosine_sim_attractions = linear_kernel(tfidf_attraction_type_matrix, tfidf_attraction_type_matrix)
```

Below is the code snippet for taking user input for location type.

```
type_entered = input("Enter the location type: ")
#type_entered = process.extract(df_attractions['location_type'],type_entered)
loc = df_attractions.loc[df_attractions['location_type'].isin([type_entered])]
location_fetch = loc.iloc[0]['location_name']
get_recommendations_based_on_type(location_fetch)
```

```
Enter the location type:
```

In this scenario, one can see that when we type Parks as our input the recommendation generates the top 2 tourist locations using the cosine similarity parameter.

```
type_entered = input("Enter the location type: ")
#type_entered = process.extract(df_attractions['location_type'],type_entered)
loc = df_attractions.loc[df_attractions['location_type'].isin([type_entered])]
location_fetch = loc.iloc[0]['location_name']
get_recommendations_based_on_type(location_fetch)
```

```
Enter the location type: Parks
Int64Index([1], dtype='int64')
2            Stanley Park
27    Kingston Waterfront
Name: location_name, dtype: object
```

The same can be said for the other features, that is province, cost, and best season to visit. Next we combined all the features together for the combined predictions.

```
[25] features = ['location_type', 'provience', 'cost','best_time_to_visit','ratings']
     #df_attractions['location_type'].isnull().values.any()
     #df_attractions['provience'].isnull().values.any()
     #df_attractions['no. of rating'].isnull().values.any()

     def combine_features(row):
         return row['location_type']+' '+row['provience']+' '+row['cost']+' '+row['best_time_to_visit']+' '+str(row['ratings'])
```

After combining all the features, we take input from the user and with that the top-3 recommended locations are suggested to the user.

Below is the code snippet for displaying how the recommendation system asks for user input.

```
location_user_likes = input("Enter a previous location you visited to help us come up with similar recommendations: ")
contect_based_combined(location_user_likes)
```

```
Enter a previous location you visited to help us come up with similar recommendations:
```

In the **snippets** below we have included the results (top-3 recommendation) from the recommendation system.

```
location_user_likes = input("Enter a previous location you visited to help us come up with similar recommendations: ")
contect_based_combined(location_user_likes)
```

```
Enter a previous location you visited to help us come up with similar recommendations: Parliament Hill and Buildings
Top 3 similar locations are
```

Fort Henry National Historic Site in the provience of ontario

Details about the attraction are specified below:

Average Rating : 4.5

Complete Address : 1 Fort Henry Drive K7L4V8, Kingston, Ontario Canada

Best time to visit in the year : all time

General visiting hours : Sun - Sat 10:00 AM - 4:30 PM

Website Link : http://www.forthenry.com/

Steveston Heritage Fishing Village in the provience of BC

Details about the attraction are specified below:

Average Rating : 4.5

Complete Address : 3820 Bayview St, Richmond, British Columbia V7E 4R7 Canada

Best time to visit in the year : all time

General visiting hours : Sat - Wed 10:00 AM - 6:00 PM

Website Link : http://www.tourismrichmond.com/things-to-do/steveston

Fort Edmonton Park in the provience of alberta

Details about the attraction are specified below:

Average Rating : 4.5

Complete Address : On Whitemud Dr, Edmonton, Alberta T5J 2R7 Canada

Best time to visit in the year : all time

General visiting hours : Tue - Sat 10:00 AM - 5:00 PM

Website Link : http://www.fortedmontonpark.ca/

Although, the system recommends the tourist location successfully, a few drawbacks/errors were encountered which will be discussed in further sections.

5.2. Clustering Model

After implementing the cosine similarity model, we tried to implement a content based clustering model where the input features to the model are location name and a short description of the location that we had. The diagram below clearly shows the process flow and steps to build a content based recommender system using K-means clustering. To begin with the clustering model, we had to choose the accurate value of k for K-means clustering algorithm. The clustering process started with preprocessing the data and modifying it as per requirements.

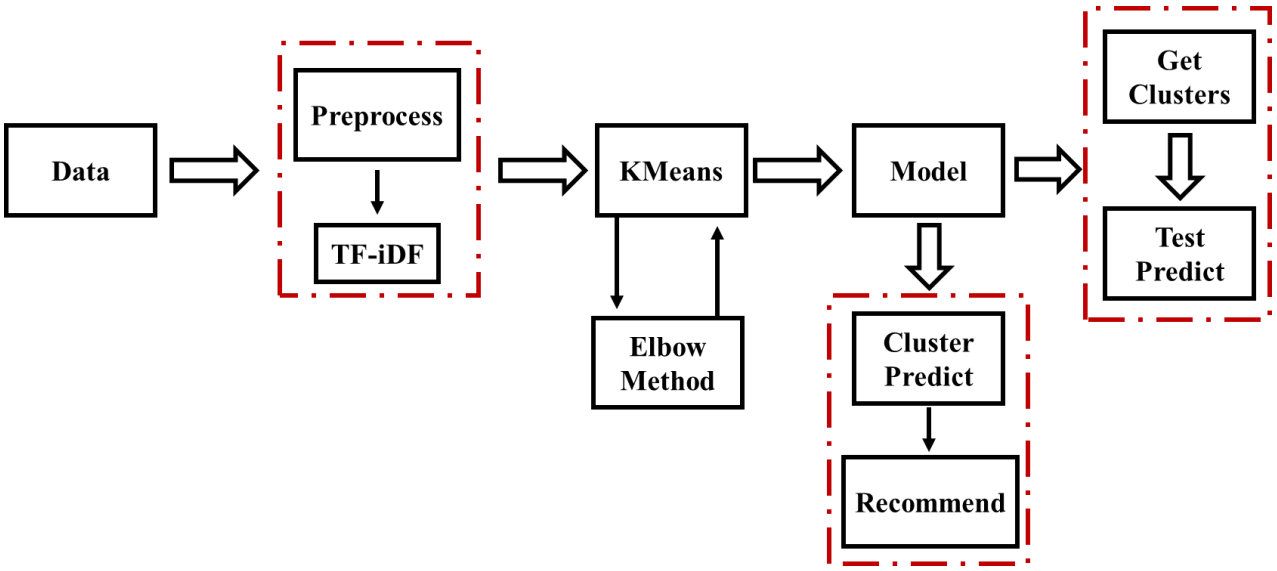


Figure 16: The process flow and steps to build a content based recommender system using K-means clustering.

The next step was to vectorized the text data that we had in hand, here we have implemented TF-iDF. The code snippet gives more on tf-idf.

```
[ ]  # Create word vectors from combined frames
     vectorizer = TfidfVectorizer(stop_words='english')
     X = vectorizer.fit_transform(comb_frame)
```

Next we used the Elbow Method for calculating the values for k.

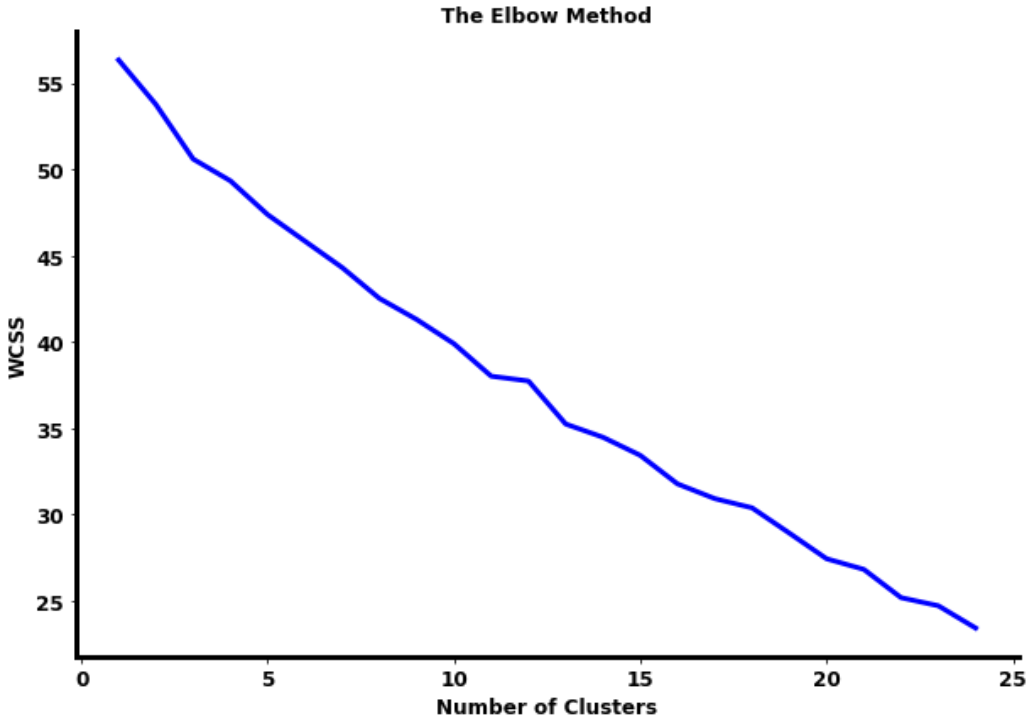


Figure 17: The implementation of the elbow method.

The graph above was generated using the elbow method, we can observe that we do not have a clear elbow point. The model will thus be trained at different flexion points to calculate and visualize the silhouette score at different number of clusters which will help choose the highest silhouette score. The next step was to implement the silhouette analysis and we have further discussed the results obtained from it and the value of k chosen after the analysis. We have also implemented silhouette analysis for multiple values of multiple clusters. Here we have included an analysis for number of cluster value as 10 and 11, 14, and 16.

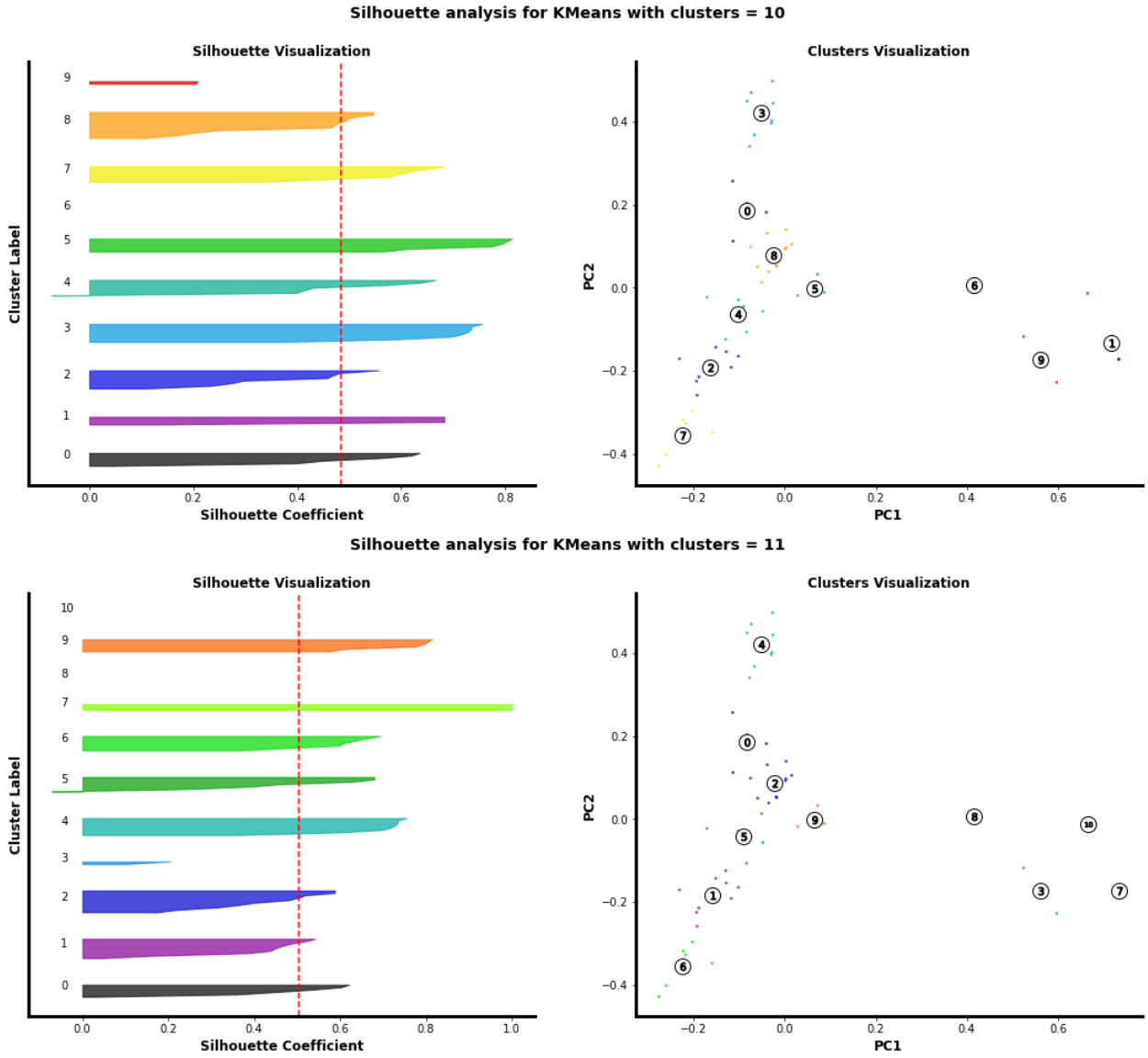


Figure 18: Shows the silhouette analysis for KMeans with cluster = 10 and 11

Here you can observe that the silhouette score gives a higher value for the number of clusters equal to 11. So, we have chosen our k value as 11. After choosing the appropriate value for k we splited our dataset in train and test set. Following that we ran the k-means algorithms and the following clusters were formed.

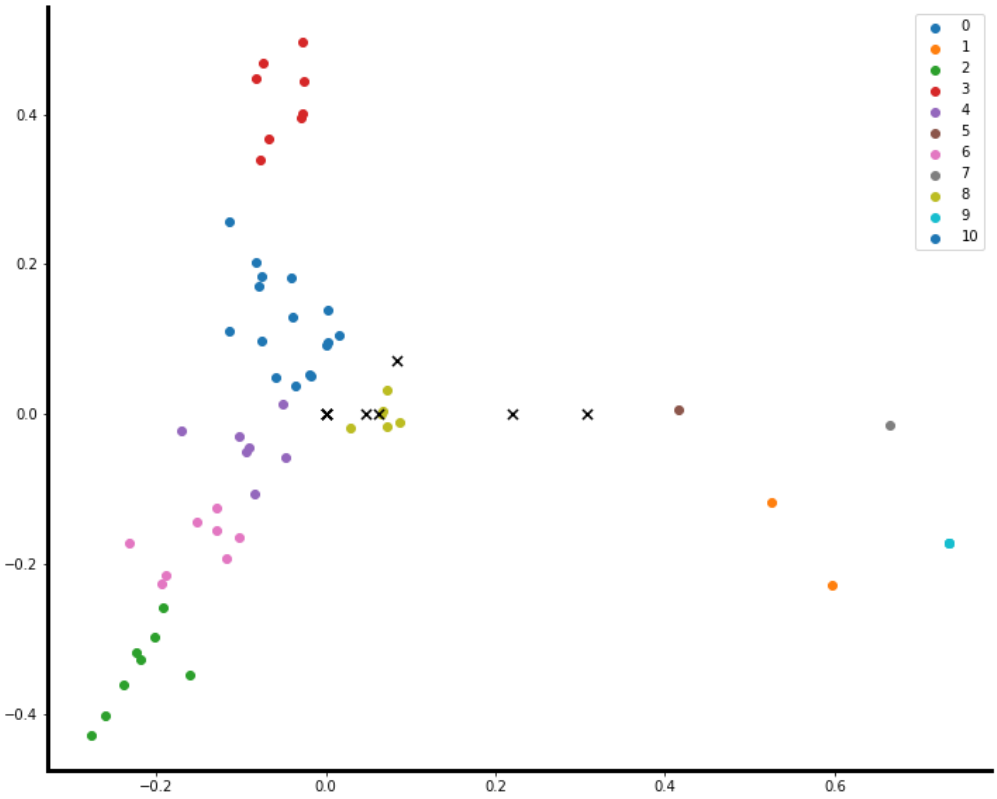


Figure 19: Shows a plot giving descriptive details on the cluster formed from the dataset.

We can observe that there are 11 clusters formed from the dataset. Next we will take a look at the contents of these clusters. The screen shot below shows top 15 words per cluster.

```
Top 15 words per cluster:

Cluster 0:
['national', 'island', 'edward', 'flowerpot', 'parks', 'park', 'islands', 'elk', 'rim', 'pacific', 'hiking', 'trails', 'prince', 'ontario', 'alberta']

Cluster 1:
['water', 'bodies', 'alberta', 'lake', 'ontario', 'athabasca', 'lost', 'maligne', 'peyto', 'moraine', 'grassi', 'lakes', 'waterfalls', 'falls', 'icefields']

Cluster 2:
['points', 'landmarks', 'ontario', 'grotto', 'riverfront', 'windsor', 'niagara', 'war', 'military', 'architectural', 'tower', 'cn', 'canada', 'basilica', 'churches']

Cluster 3:
['museums', 'speciality', 'museum', 'canadian', 'history', 'bc', 'royal', 'western', 'saskatchewan', 'development', 'human', 'rights', 'manitoba', 'quebec', 'landmarks']

Cluster 4:
['historic', 'sites', 'village', 'heritage', 'fort', 'buildings', 'ontario', 'alberta', 'dundurn', 'castles', 'castle', 'parks', 'historical', 'park', 'amusement']

Cluster 5:
['mountains', 'whistler', 'bc', 'blackcomb', 'mountain', 'sports', 'complexes', 'ski', 'snowboard', 'bike', 'areas', 'park', 'heritage', 'henry', 'highway']

Cluster 6:
['bc', 'park', 'parks', 'canyon', 'bridges', 'stanley', 'lynn', 'point', 'neck', 'trestle', 'kinsol', 'provincial', 'brandywine', 'canyons', 'state']

Cluster 7:
['quebec', 'areas', 'old', 'montreal', 'tremblant', 'ski', 'snowboard', 'zoo', 'zoos', 'calgary', 'walking', 'nature', 'wildlife', 'neighbourhoods', 'points']

Cluster 8:
['nova', 'scotia', 'gardens', 'halifax', 'valley', 'annapolis', 'valleys', 'novascotia', 'boardwalk', 'boardwalks', 'piers', 'waterfront', 'public', 'henry', 'grassi']

Cluster 9:
['ontario', 'parks', 'park', 'gage', 'kingston', 'waterfront', 'mount', 'toronto', 'royal', 'mills', 'vaughan', 'islands', 'springbank', 'ripleys', 'aquariums']

Cluster 10:
['trams', 'banff', 'gondola', 'alberta', 'grotto', 'historic', 'hill', 'hiking', 'highway', 'heritage', 'henry', 'halifax', 'government', 'grassi', 'history']
```

We can see how the clusters have been formed and each cluster has a wide range of token words in them which can be helpful to predict the cluster of the new words/descriptions about the attraction. Based on these clusters we will try to fit the new word in the existing cluster and then provide a corresponding recommendation to the user. In the screen shot shown below we can see how the text for location type is being clustered into different clusters and the predicted cluster value can be observed.

location_id	location_name	location_type	ratings	no. of rating	provience	user_id	Input_Text	Predicted_Clusters
17	Port Credit	Bodies of Water Neighbourhoods	4.5	953	ontario	2	Port Credit Bodies of Water Neighbourhoods o...	1
11	West Edmonton Mall	Shopping Malls	4.0	4,198	alberta	4	West Edmonton Mall Shopping Malls alberta	1
44	Elk Island National Park	National Parks	2.5	600	alberta	4	Elk Island National Park National Parks alberta	0
49	Lynn Canyon Park	Parks	3.0	2,174	BC	5	Lynn Canyon Park Parks BC	6
47	Sea to Sky Highway	Scenic Drives	4.5	1,492	BC	5	Sea to Sky Highway Scenic Drives BC	6

Next we will make recommendations based on the clustering results. The recommendation that the model has generated can be seen in the screen shot shown below.

```
We have 5 recommenations for [Mount Royal Park]:
['Toronto Island Park', 'Kingston Waterfront', 'Mount Royal Park', "Ripley's Aquarium of Canada", 'Springbank Park']

We have 5 recommenations for [Old Quebec]:
['Old Quebec', 'Tremblant', 'Old Montreal', 'Le Petit Train du Nord Bike Path', 'The Calgary Zoo']

We have 5 recommenations for [Niagara Falls Canada]:
['Niagara Falls Canada', 'Canadian War Museum', 'Signal Hill', 'Windsor Riverfront', 'The Grotto']
```

5.3. Classification Model(using scikit learn library)

To draw conclusions from the dataset, we have used three classification models to assign tags to text according to its content. We have first implemented a classification on the basis of user id using

- Support vector machines
- K-Nearest Neighbours
- Decision tree

We have further used the precision, recall, f1-score and support metrics to classify the best model. Performing these algorithms involved cleaning the data by removing punctuations, multiple spaces, stop words and converting to lower cases. After performing the data cleaning, we then divides our dataset into train and test dataset. Code snippet below shows how data is divided.

```
from sklearn.model_selection import train_test_split
df_attractions_train, df_attractions_test, df_attractions_userId_train, df_attractions_userId_test = train_test_split(
df_clean['Names'],df_clean['User_id'], test_size=0.15, random_state=0)
```

For feature engineering, the dataset was then put into vector space using TF-iDF vectorizer. Using the sci-kit learn library this function was used to sum the number of word occurrences in the dataset with the number of times it appears in the dataset, it thus shows the frequency of words in the dataset. The vectorized data is then being used for different algorithms.

Classification based on user Id as the input labels

Support Vector Machine(SVM):

To analyze the data and detect outliers this model was used to check the model parameters and implement the baseline model and also measure its accuracy. As much as this model works best with small dataset it is good for data containing text images. The model performance will further be analyzed based on the test data using visualization. The accuracy for the SVM base model has been calculated as shown below. From the code snippet below, the accuracy score for the SVM is **0.2**

```
# Check the accuracy of the SVM base model:
from sklearn import svm
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
SVM_base_model = svm.SVC()
SVM_base_model.fit(df_attractions_train_tfidf, df_attractions_userId_train)
SVM_base_model_accur=accuracy_score(df_attractions_userId_test, SVM_base_model.predict(df_attract
print('The accuracy score of SVM_base_model:',SVM_base_model_accur)
```

```
The accuracy score of SVM_base_model: 0.2222222222222222
```

K-Nearest Neighbour:

This model classifies data by getting the important center nodes. The data is classified by calculating the means of the data and assigning them to the class closest to their mean.

```
from sklearn.neighbors import KNeighborsClassifier
KNN=KNeighborsClassifier()
print('Check the parameters of KNN:\n',KNN.get_params())
```

```
Check the parameters of KNN:
 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}
```

```
KNN_base_model = KNeighborsClassifier()
KNN_base_model.fit(df_attractions_train_tfidf, df_attractions_userId_train)
KNN_base_model_accur=accuracy_score(df_attractions_userId_test, KNN_base_model.predict(df_attractions_test_tfidf))
print('The accuracy score of KNN_base_model:',KNN_base_model_accur)
```

```
The accuracy score of KNN_base_model: 0.1111111111111111
```

From **shots** above, the accuracy score of KNN model is **0.11**

Decision Tree:

This model is a supervised learning method used in classification, it predicts the value of a target variable by breaking down the complexity of the problem into branches.

```
# DT base model:
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

DT_base_model= DecisionTreeClassifier() # Create Decision Tree classifer object
DT_base_model=DT_base_model.fit(df_attractions_train_tfidf, df_attractions_userId_train) # Train Decision Tree Classifer
#Predict the response for test dataset
DT_base_model_pred = DT_base_model.predict(df_attractions_test_tfidf)
DT_base_model_accur=accuracy_score(df_attractions_userId_test, DT_base_model.predict(df_attractions_test_tfidf))
print("Accuracy:",metrics.accuracy_score(df_attractions_userId_test, DT_base_model_pred))
```

```
Accuracy: 0.2222222222222222
```

From **shot** above, the decision tree classifier was passed to the training data and user id in vector planes and the accuracy score is **0.2.**

In conclusion, from all the analysis shown, the SVM is our chosen champion model and it predicts the user id based on the results shown above. To analyze the performance of the model on the test data, a confusion matrix and classification report will be shown below.

Error Analysis:

The champion model was classified on the basis of

Precision: This is the ratio of $t_p/(t_p+f_p)$. It helps as a classifier that does not label the positive sample as negative.

Recall: It is the ratio of $t_p/(t_p+f_n)$ where it finds all the positive samples.

F1-score: Is the weighted harmonic mean of the precision and recall where its best value is 1 and its worst score is 0.

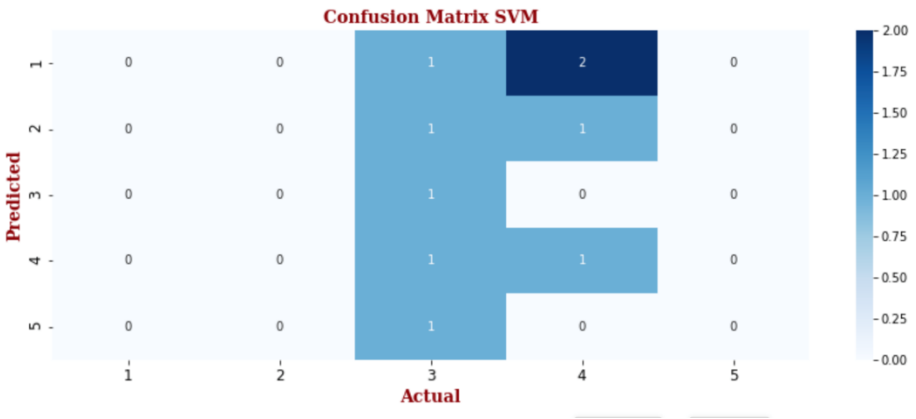
Support: This is simply the number of occurrences of each class of y_true.

Classification report of SVM model:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	3
2	0.00	0.00	0.00	2
3	0.20	1.00	0.33	1
4	0.25	0.50	0.33	2
5	0.00	0.00	0.00	1
accuracy			0.22	9
macro avg	0.09	0.30	0.13	9
weighted avg	0.08	0.22	0.11	9

Confusion matrix for SVM

Confusion Matrix of SVM model:



From the confusion matrix generated for the SVM model we can make out that the user ids are evenly distributed. This shows a correct prediction using the SVM with an evenly random split of the dataset. The graph below helps us understand the model predictions based on user_id as label. We have shown how the predicted and actual user_id are. Here we have tried to make our graph interactive so on hover on the bar plots information regarding its label values can be gotten.

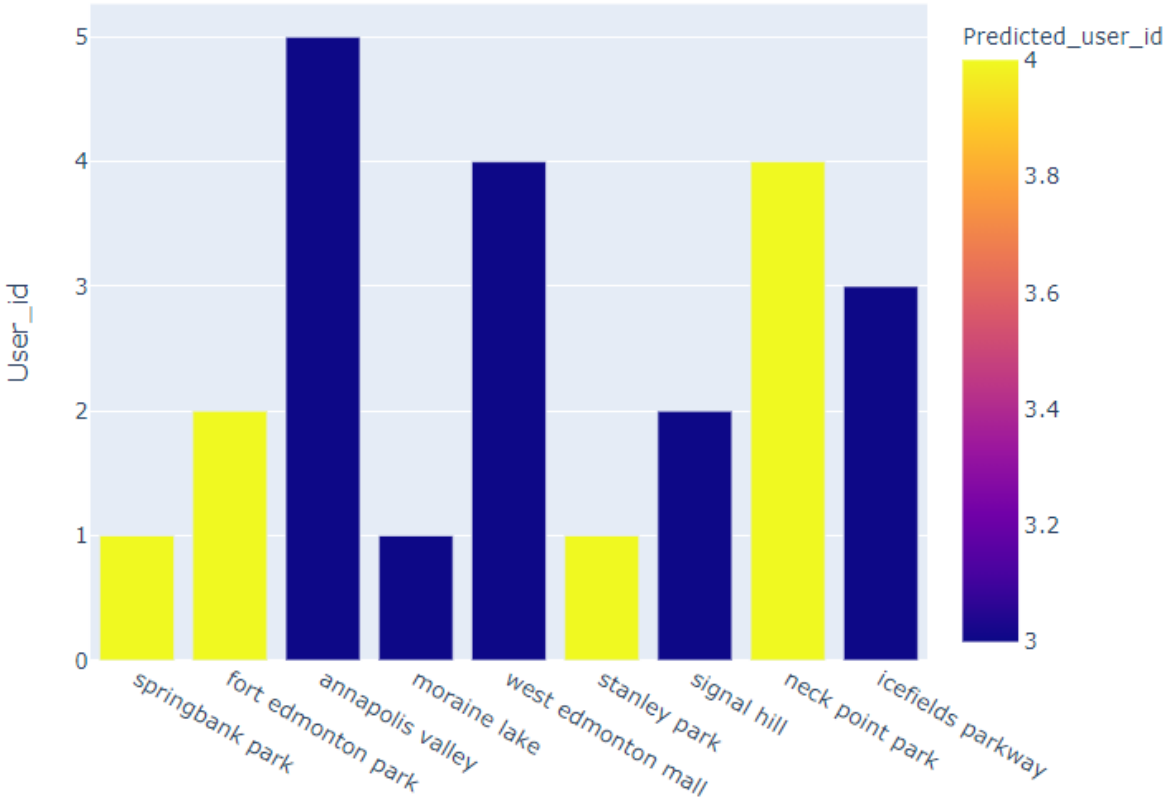


Figure 20: Model predictions based on user_id as label

Classification on the basis of Location Type as input labels:

We follow the same procedure for text cleaning and text vectorization as done for user_Id as labels. Again, we have used KNN, SVM and Decision tree to evaluate our results and choose which algorithms perform best for the classification model.

In this section we check:

Support Vector Machines:

```
# Check the accuracy of the SVM base model:
from sklearn import svm
SVM_base_model = svm.SVC()
SVM_base_model.fit(df_attractions_train_tfidf, df_attractions_typeId_train)
SVM_base_model_accur=accuracy_score(df_attractions_typeId_test, SVM_base_model.predict(df_attractions_test_tfidf))
print('The accuracy score of SVM_base_model:',SVM_base_model_accur)
```

```
The accuracy score of SVM_base_model: 0.4444444444444444
```

From **shot 21** above, the SVM library is used and then the accuracy score based on location type is calculated. It shows the score is **0.44**

K-Nearest Neighbour: It shows the score is **0.44** as previously shown using the SVM algorithm.

```
KNN_base_model = KNeighborsClassifier()
KNN_base_model.fit(df_attractions_train_tfidf, df_attractions_typeId_train)
KNN_base_model_accur=accuracy_score(df_attractions_typeId_test, KNN_base_model.predict(df_attractions_test_tfidf))
print('The accuracy score of KNN_base_model:',KNN_base_model_accur)
```

```
The accuracy score of KNN_base_model: 0.4444444444444444
```

Decision Tree: shows the score of **0.44**

```
# DT base model:
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

DT_base_model= DecisionTreeClassifier() # Create Decision Tree classifer object
DT_base_model=DT_base_model.fit(df_attractions_train_tfidf, df_attractions_typeId_train) # Train Decision Tree Classifer
#Predict the response for test dataset
DT_base_model_pred = DT_base_model.predict(df_attractions_test_tfidf)
DT_base_model_accur=accuracy_score(df_attractions_typeId_test, DT_base_model.predict(df_attractions_test_tfidf))
print("Accuracy:",metrics.accuracy_score(df_attractions_typeId_test, DT_base_model_pred))
```

```
Accuracy: 0.4444444444444444
```

Error Analysis:

So next step was to calculate the classification report for all the algorithms
The classification report for each algorithm can be in the images attached below.

KNN:

Classification report of KNN model:

	precision	recall	f1-score	support
2	0.75	1.00	0.86	3
4	0.00	0.00	0.00	1
5	0.20	1.00	0.33	1
6	0.00	0.00	0.00	1
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
accuracy			0.44	9
macro avg	0.14	0.29	0.17	9
weighted avg	0.27	0.44	0.32	9

SVM:

Classification report of SVM model:

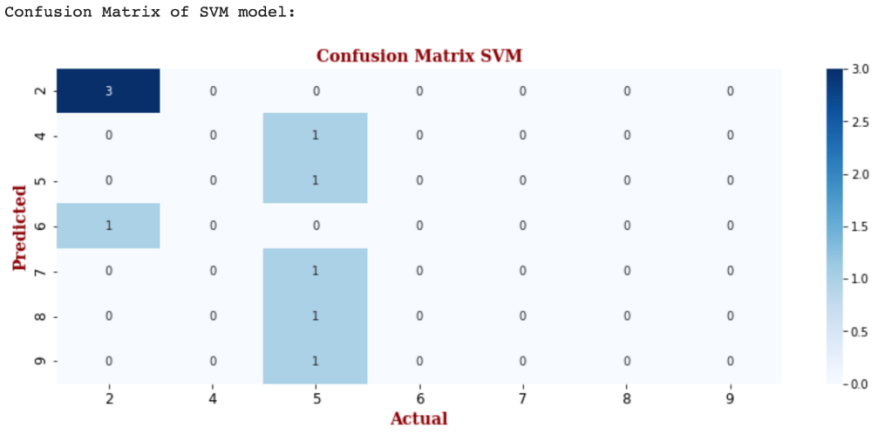
	precision	recall	f1-score	support
2	0.75	1.00	0.86	3
4	0.00	0.00	0.00	1
5	0.20	1.00	0.33	1
6	0.00	0.00	0.00	1
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
accuracy			0.44	9
macro avg	0.14	0.29	0.17	9
weighted avg	0.27	0.44	0.32	9

Decision Tree:

Classification report of DT model:

	precision	recall	f1-score	support
2	0.75	1.00	0.86	3
4	0.00	0.00	0.00	1
5	0.20	1.00	0.33	1
6	0.00	0.00	0.00	1
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
accuracy			0.44	9
macro avg	0.14	0.29	0.17	9
weighted avg	0.27	0.44	0.32	9

After checking the accuracy and classification report for all the model, we have chosen SVM as our champion model in this case as well. The confusion matrix for our best model is as shown in **below.**



Here the confusion matrix helps us to show how the classification models performed. We can observe from the confusion matrix that most of the data is either predicted as location type1 or as location type4, which gives us about 44% of accuracy.
The bar plot chart will help us understand the SVM classification prediction labels in more detail.

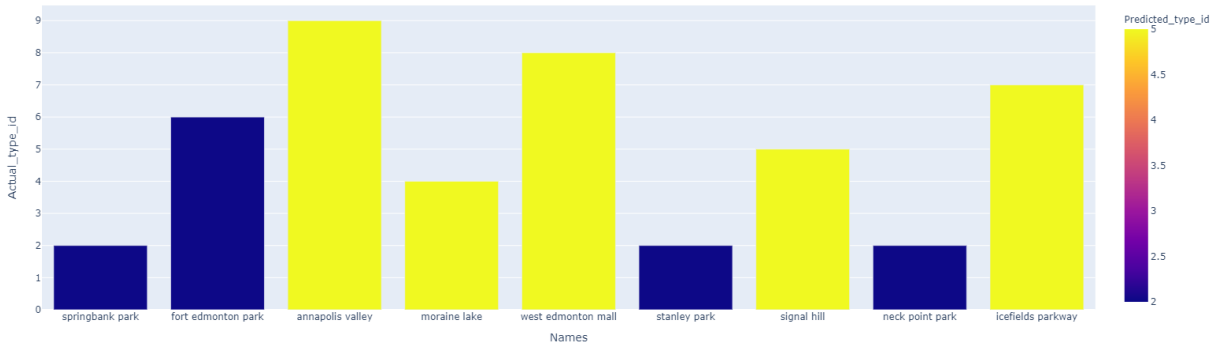


Figure 21: Bar plot showing the SVM classification prediction labels

5.4. Classification model (surprise library)

We implemented collaborative filtering for the recommendation system and in this case, the surprise library was used. Since the system aims to predict user preference on past experience by filtering the users preference on the basis of past habits of similar users.
Collaborative filtering has two types of algorithms:

- **Memory based algorithms:**

This algorithm applies statistical techniques to the dataset in order to calculate predictions and it is divided into two main sections (user-item filtering and item-item filtering). Here the closest users or items are calculated only by using cosine similarity or Pearson correlation coefficients, and these are solely based on arithmetic operations.

- **Model based algorithms:**

In the case of the model based algorithms, it involves building machine learning algorithms that predict the user's rating. This thus involves using dimensionality reduction methods that reduces the high dimensional matrix containing so many missing values with a much smaller matrix in lower dimensional space.
In our project, we introduced both algorithms and compared them by performing cross validation with an evaluation based on RSME and MAE to choose the best algorithm.
These evaluation metrics were used to get the difference between the estimated and actual value of the algorithms mentioned above and also to get the difference between the original predicted values over the dataset that were extracted by the squared average difference respectively. After these processes, the results were further explored.

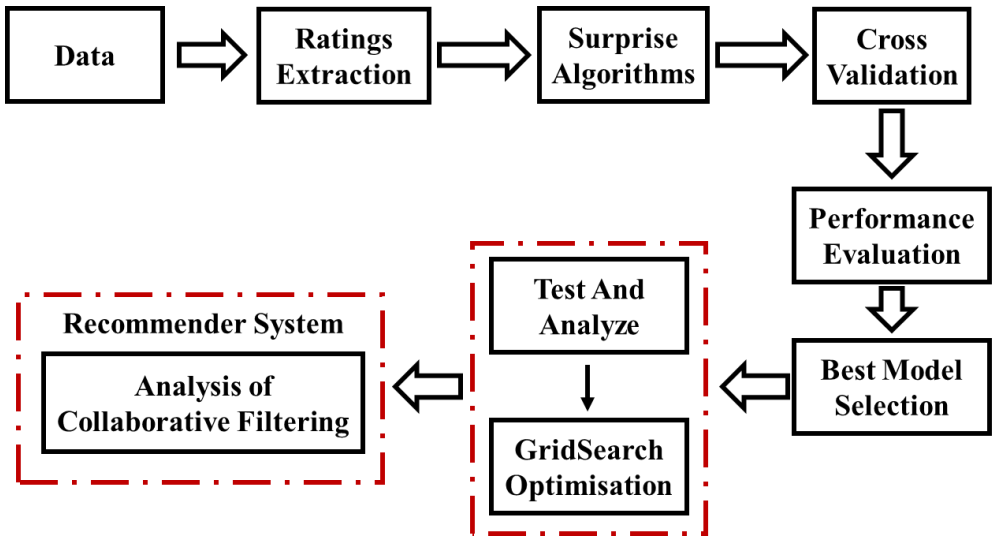


Figure 22: Flowchart to represent the recommender system via collaborative filtering analysis.

As mentioned earlier, we have used surprised library and other relevant libraries such as NumPy, matplotlib, seaborn and pandas to achieve this implementation. The dataset was loaded into Google Collaboratory using pandas data frame and it was further broken into a subset of the following columns (**location_id, type_id, rating and no. of rating**) for the visualizations shown below:

	location_id	type_id	ratings	no. of rating
0	0	4	4.5	20,326
1	1	2	3.0	10,493
2	2	2	4.5	28,722
3	3	4	1.0	31,476
4	4	5	5.0	13,360

Shot above shows a subset of the main data set.
Before implement the classification algorithms, we tried to visualize the subset of the main dataset. The images shown below will help us understand the subset clearly.

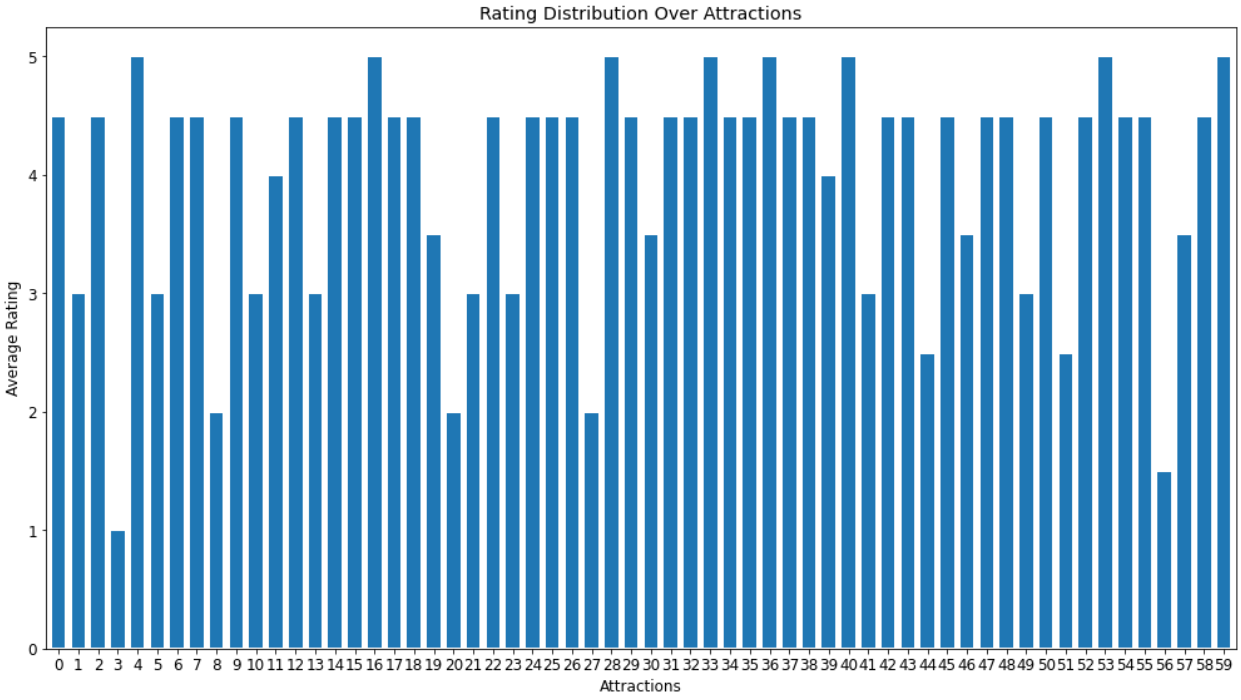


Figure 23: Visualization results of the subset of main dataset.

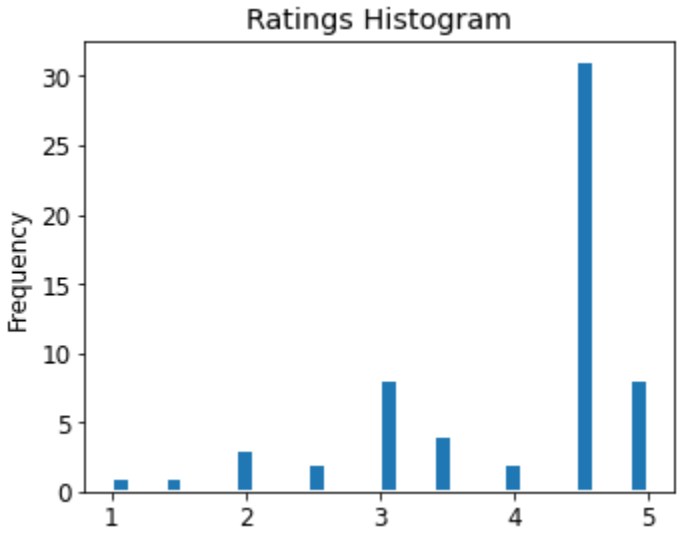


Figure 24: Histogram showing the number of average ratings distribution over attractions.

Cross Validation:

Cross validation was used to compute the error for the various model offered within the surprise library. Surprise library offers variety of algorithm model for classification and we have tried to use the following algorithms for our classification model.

The algorithms used are

1. SVD()
2. SVDpp()
3. SlopeOne()
4. NMF()
5. NormalPredictor()
6. KNNBaseline()
7. KNNBasic()
8. KNNWithMeans()
9. KNNWithZScore
10. BaselineOnly()
11. CoClustering()

In this sections, we compared these models and decided on the best performance based on RSME(Root Mean Squared Error).

```
benchmark = []
# Iterate over all algorithms
for algorithm in [SVD(), SVDpp(), SlopeOne(), NMF(), NormalPredictor(),
                  KNNBaseline(), KNNBasic(), KNNWithMeans(), KNNWithZScore(),
                  BaselineOnly(), CoClustering()]:
    # Perform cross validation
    results = cross_validate(algorithm, data, measures=['RMSE','MAE'], cv=3, verbose=False,return_train_measures=True)
    # Get results & append algorithm name
    tmp = pd.DataFrame.from_dict(results).mean(axis=0)
    tmp = tmp.append(pd.Series([str(algorithm).split(' ')[0].split('.')[-1]], index=['Algorithm']))
    benchmark.append(tmp)
```

```
print('3 Fold Cross Validation Result sorted based on Test_RMSE (ascending)\n')
surprise_results = pd.DataFrame(benchmark).set_index('fit_time').sort_values('test_rmse', ascending=True)
surprise_results
```

```
3 Fold Cross Validation Result sorted based on Test_RMSE (ascending)
```

fit_time	test_rmse	train_rmse	test_mae	train_mae	test_time	Algorithm
0.004076	0.971618	0.000000	0.812083	0.000000	0.000216	KNNWithZScore
0.000167	0.973790	0.000000	0.812917	0.000000	0.000230	KNNBasic
0.006128	0.974338	0.168135	0.812083	0.132816	0.000238	NMF
0.000376	0.974649	0.000000	0.816250	0.000000	0.000165	KNNWithMeans
0.003369	0.982302	0.693551	0.819130	0.572815	0.000211	SVD
0.000235	0.989690	0.200111	0.830000	0.163826	0.000215	SlopeOne
0.012379	0.991510	0.479026	0.828750	0.389883	0.000341	CoClustering
0.000374	0.996201	0.858753	0.825287	0.715221	0.000165	BaselineOnly
0.002574	1.001411	0.000000	0.839494	0.000000	0.000193	KNNBaseline
0.008206	1.005306	0.705921	0.841459	0.589317	0.000153	SVDpp
0.000108	1.403521	1.308615	1.177690	1.056864	0.000216	NormalPredictor

Show above shows the code snippet and result represents the surprise results for RMSE cross validation test in ascending order.

After computing the cross validation ,we printed the values of each model in ascending order.

```
print('3 Fold Cross Validation Result sorted based on fit_time (ascending)\n')
#surprise_results = pd.DataFrame(benchmark).set_index('fit_time').sort_values('fit_time', ascending=True)
surprise_results = pd.DataFrame(benchmark).sort_values('fit_time', ascending=True)
surprise_results
```

```
3 Fold Cross Validation Result sorted based on fit_time (ascending)
```

	test_rmse	train_rmse	test_mae	train_mae	fit_time	test_time	Algorithm
4	1.403521	1.308615	1.177690	1.056864	0.000108	0.000216	NormalPredictor
6	0.973790	0.000000	0.812917	0.000000	0.000167	0.000230	KNNBasic
2	0.989690	0.200111	0.830000	0.163826	0.000235	0.000215	SlopeOne
9	0.996201	0.858753	0.825287	0.715221	0.000374	0.000165	BaselineOnly
7	0.974649	0.000000	0.816250	0.000000	0.000376	0.000165	KNNWithMeans
5	1.001411	0.000000	0.839494	0.000000	0.002574	0.000193	KNNBaseline
0	0.982302	0.693551	0.819130	0.572815	0.003369	0.000211	SVD
8	0.971618	0.000000	0.812083	0.000000	0.004076	0.000216	KNNWithZScore
3	0.974338	0.168135	0.812083	0.132816	0.006128	0.000238	NMF
1	1.005306	0.705921	0.841459	0.589317	0.008206	0.000153	SVDpp
10	0.991510	0.479026	0.828750	0.389883	0.012379	0.000341	CoClustering

Visualization of results

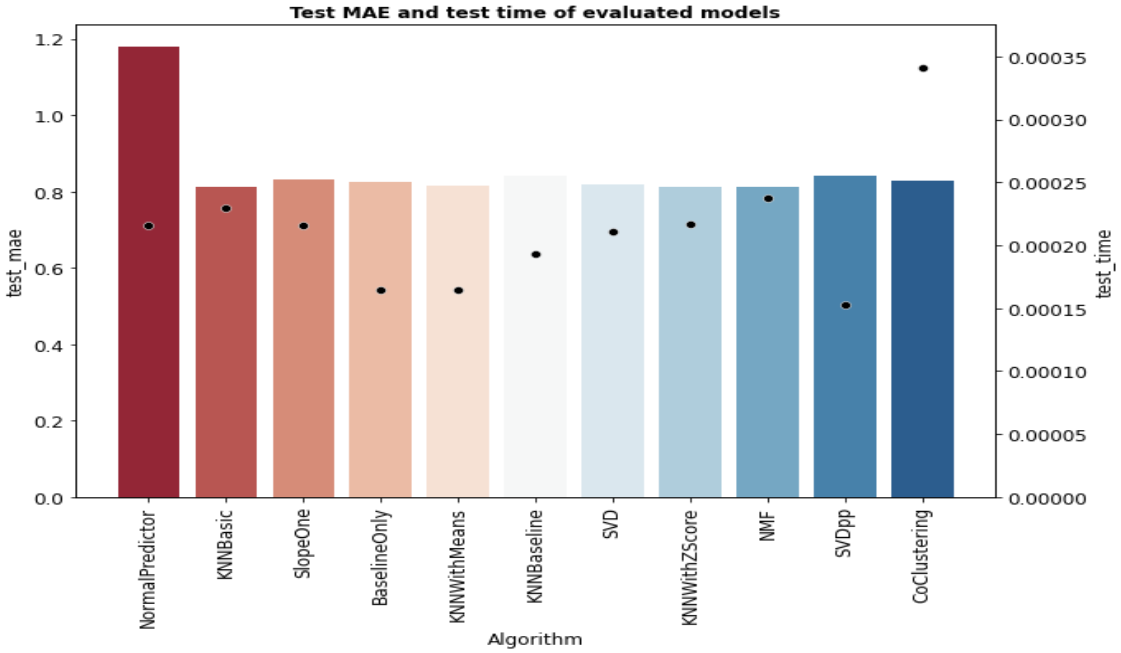


Figure 25: The test time and mean absolute error of all the evaluated models.

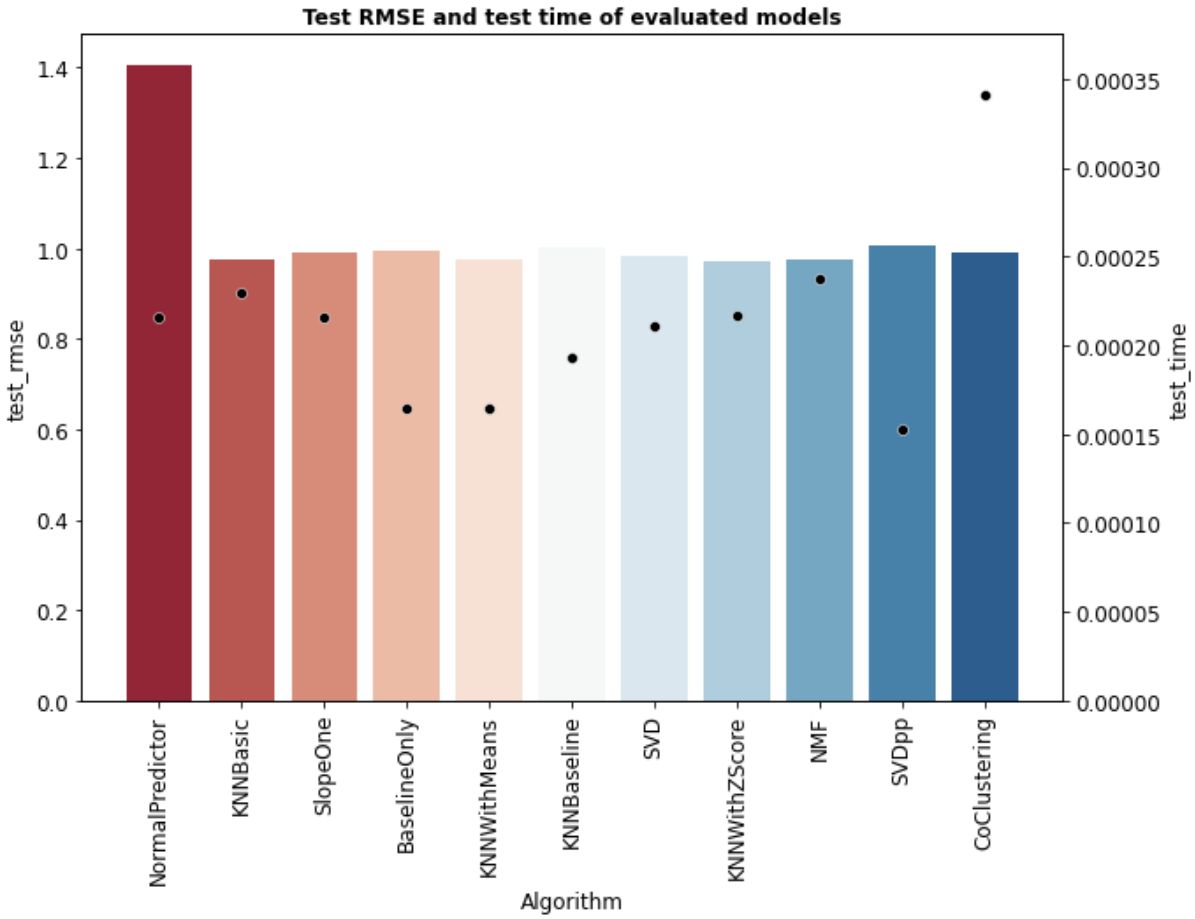


Figure 26: The test time and Root Mean Square error of all the evaluated models.

K-Nearest Neighbors Model selection

Surprise library offers variations of this non parametric classification algorithm. It calculates the distance between users or items and finds the closet one that are equivalent to the most similar ones. With the comparison of the model variations it shows that KNNwithZScore algorithm gives the best performance of RSME.

Item-Item Model Selection

This approach is used to determine a set of the most similar items to the items on the basis of past rating of other users and it also calculates the rating for the item based on the rating of similar items rated by the user.

From the analysis, Pearson correlation coefficient gives the best results in terms of RSME and MAE; it thus shows that using this approach proves much improvement compared to the previous iteration discussed above.

	Algorithm	fit_time	test_mae	test_rmse	test_time
2	KNNWithZScore pearson	0.003642	0.807083	0.968414	0.000216
0	KNNWithZScore cosine	0.003781	0.833750	0.971889	0.000217
1	KNNWithZScore msd	0.002779	0.850417	0.994243	0.000186

Shot above shows the similarities for KNNwithZScore algorithm (for Pearson, cosine and msd) and results of the computation.

We also tried to visualize the above obtained results in a plot as seen below.

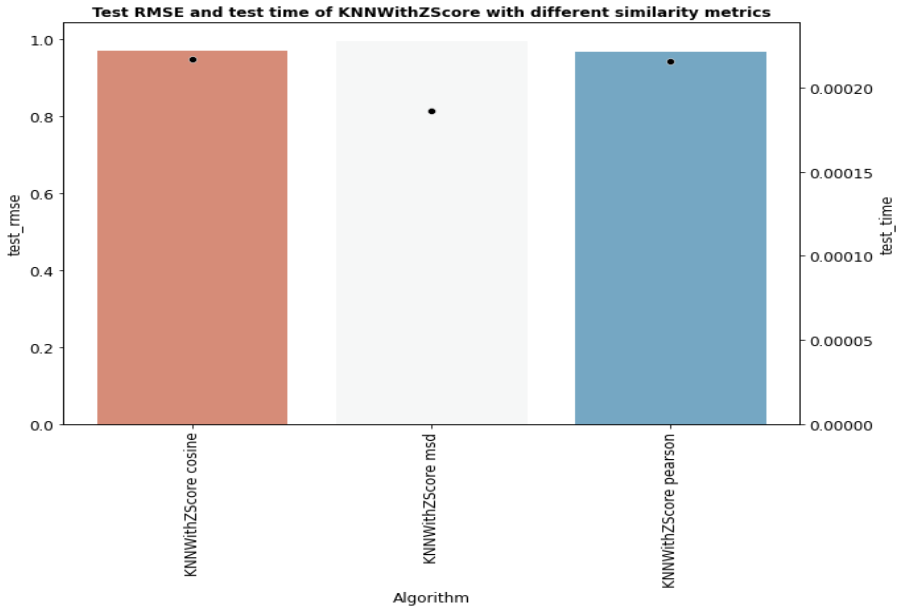


Figure 27: The the results for the test time of KNNwithZScore and the test RMSE for different similarity metrics.

KNN with ZScore Model Optimization with Grid search Cross validation:

We have further implemented a grid search cross validation with the KNNwithZScore model this helps to compute the accuracy metrics for an algorithm with various combinations of parameters. It is thus useful for finding the best configuration of parameters. From our analysis, the best set of parameters in terms of RSME obtained using grid search cross validation is shown below.

- KNNwithZScore, item-item, Pearson similarity metric
- K = 40
- Min_k = 1
- Min_support = 1
- RSME = 0.9692251202388923

Error Analysis of Collaborative Filtering model results:

This section gives a detailed description of the results obtained by the KNN model with the best RMSE score.

```
trainset, testset = train_test_split(data, test_size=0.25)
sim_options = {'name': 'pearson','user_based': False,'min_support': 1}
model = KNNWithZScore(k=40, min_k=1, sim_options=sim_options)
predictions = model.fit(trainset).test(testset)
accuracy.rmse(predictions)
print(model.__class__.__name__)
```

```
Computing the pearson similarity matrix...
Done computing similarity matrix.
RMSE: 0.7106
KNNWithZScore
```

Shot is a code snippet that shows the results of the pearson similarity matrix on the KNNwithZScore model with the test and trained data given the parameters for k, min_k, and min_support.

```
def k_from_details(details):
    try:
        return details['actual_k']
    except KeyError:
        return 1000
```

```
df_pred = pd.DataFrame(predictions, columns=['user_id', 'location_id', 'actual_rating', 'pred_rating', 'details'])
df_pred['k'] = df_pred['details'].apply(k_from_details)
df_pred['impossible'] = df_pred['details'].apply(lambda x: x['was_impossible'])
df_pred['pred_rating_round'] = df_pred['pred_rating'].round()
df_pred['abs_err'] = abs(df_pred['pred_rating'] - df_pred['actual_rating'])
df_pred.drop(['details'], axis=1, inplace=True)
df_pred.sample(5)
```

	user_id	location_id	actual_rating	pred_rating	k	impossible	pred_rating_round	abs_err
0	2	2	4.5	3.888889	1000	True	4.0	0.611111
11	7	33	5.0	3.888889	1000	True	4.0	1.111111
8	7	13	3.0	3.888889	1000	True	4.0	0.888889
7	2	12	4.5	3.888889	1000	True	4.0	0.611111
3	5	22	4.5	3.888889	1000	True	4.0	0.611111

Shot below is code snippet showing the predictions ratings and absolute error

```
best_predictions = df_pred.sort_values(by='abs_err')[:10]
best_predictions
```

	user_id	location_id	actual_rating	pred_rating	k	impossible	pred_rating_round	abs_err
12	8	39	4.0	3.888889	1000	True	4.0	0.111111
2	5	57	3.5	3.888889	1000	True	4.0	0.388889
0	2	2	4.5	3.888889	1000	True	4.0	0.611111
1	1	24	4.5	3.888889	1000	True	4.0	0.611111
3	5	22	4.5	3.888889	1000	True	4.0	0.611111
4	5	34	4.5	3.888889	1000	True	4.0	0.611111
5	5	6	4.5	3.888889	1000	True	4.0	0.611111
6	7	47	4.5	3.888889	1000	True	4.0	0.611111
7	2	12	4.5	3.888889	1000	True	4.0	0.611111
13	2	58	4.5	3.888889	1000	True	4.0	0.611111

```
worst_predictions = df_pred.sort_values(by='abs_err')[-10:]
worst_predictions
```

	user_id	location_id	actual_rating	pred_rating	k	impossible	pred_rating_round	abs_err
4	5	34	4.5	3.888889	1000	True	4.0	0.611111
5	5	6	4.5	3.888889	1000	True	4.0	0.611111
6	7	47	4.5	3.888889	1000	True	4.0	0.611111
7	2	12	4.5	3.888889	1000	True	4.0	0.611111
13	2	58	4.5	3.888889	1000	True	4.0	0.611111
14	1	15	4.5	3.888889	1000	True	4.0	0.611111
8	7	13	3.0	3.888889	1000	True	4.0	0.888889
9	5	21	3.0	3.888889	1000	True	4.0	0.888889

The snippets above shows code and results of the best and worst predictions as per absolute error. We further plotted a distribution of actual and predicted ratings in the test data and from the distribution of actual ratings of attractions in the test set, the biggest part of the users give positive scores of about 4.5.

Similarly, we discovered that the distribution of predicted ratings in the test set is visibly different and therefore shows that the recommender system is not perfect and it cannot reflect the real distribution of attraction ratings.

```
palette = sns.color_palette("RdBu", 10)
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(15, 7))
sns.countplot(x='actual_rating', data=df_pred, palette=palette, ax=ax1)
ax1.set_title('Distribution of actual ratings of attractions in the test set')
sns.countplot(x='pred_rating_round', data=df_pred, palette=palette, ax=ax2)
ax2.set_title('Distribution of predicted ratings of attractions in the test set')
plt.show()
```

The code snippet produces a plot that displays the distribution of actual and predicted ratings of attractions in the test set.

According to the distribution of actual ratings of attractions in the test set, the biggest part of users give positive scores (around 4.5). The distribution of predicted ratings in the test set is visibly different (4). It shows that the recommender system is not perfect, and it cannot reflect the real distribution of attraction ratings.

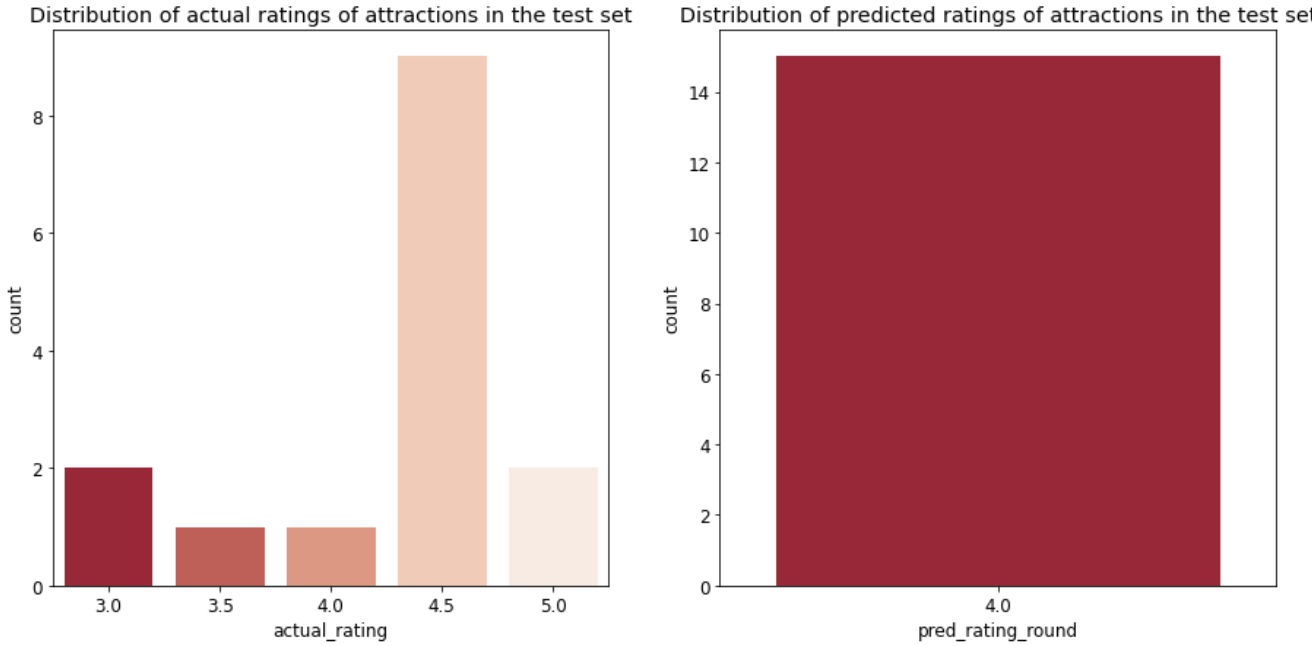


Figure 28: Dsitibution of actual ratings and predicted ratings.

Absolute error of predicted ratings:

The distribution of absolute errors is centered, showing that the majority of errors is small: between -0.25 and 1.5. As expected from the previous charts, the model deals very well with predicting score = 4.5 (the most frequent value). The further the rating from score = 4.5, the higher the absolute error. The biggest errors happen to observations with scores 2 or 3 which indicates that probably the model is predicting high ratings for those observations.

```
df_pred_err = df_pred.groupby('actual_rating')['abs_err'].mean().reset_index()
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(15, 7))
sns.distplot(df_pred['abs_err'], color='#2f6194', ax=ax1)
ax1.set_title('Distribution of absolute error in test set')
sns.barplot(x='actual_rating', y='abs_err', data=df_pred_err, palette=palette, ax=ax2)
ax2.set_title('Mean absolute error for rating in test set')
plt.show()
```

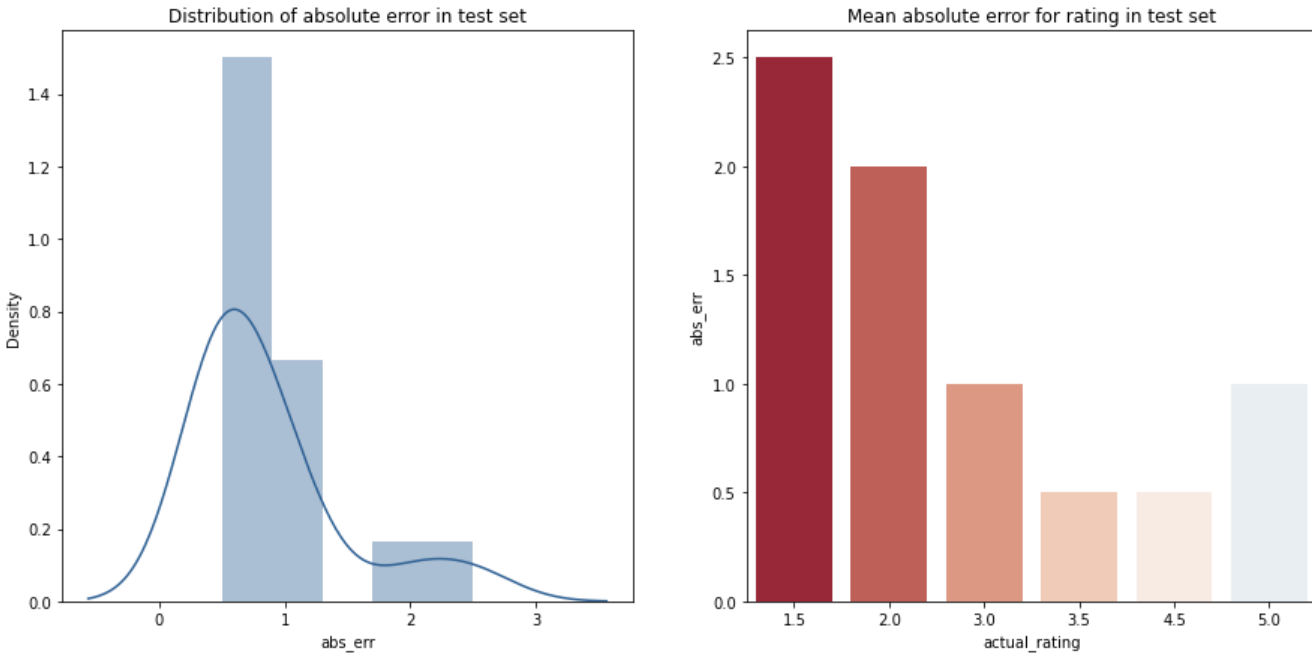


Figure 29: Dsitibution of absolute error in test setand mean absolute error for actual ratings.

These analysis were further tested on the system with the set of spitted train sets in order to get all predictions. From the first train set, the predictions was used to get the accuracy of the RSME predictions with values of k and min_k = 40 and 1 respectively.

```
Computing the pearson similarity matrix...
Done computing similarity matrix.
RMSE: 0.7106
0.710590086903088
```

From that the RSME value can be seen to be **0.7106**

```
#To get top 5 reommendation
n = 5
for uid, user_ratings in all_pred.items():
    user_ratings.sort(key=lambda x: x[1], reverse=True)
    all_pred[uid] = user_ratings[:n]
```

```
#tmp = pd.DataFrame.from_dict(all_pred,orient='index')
tmp = pd.DataFrame.from_dict(all_pred)
tmp_transpose = tmp.transpose()
```

```
def get_predictions(user_id):
    results = tmp_transpose.loc[user_id]
    return results
```

```
#specifying the user
user_id= 4
results = get_predictions(user_id)
results
```

The snippet above gets the top 5 recommendation for the uid, user_ratings in all the predicted items and then transposes the matrix, same was done similarly for the user id with the results done shown below

```
0     (40, 5)
1     (36, 5)
2     (16, 5)
3      (4, 5)
4    (9, 4.5)
Name: 4, dtype: object
```

```
recommended_attraction_ids=[]
for x in range(0, n):
    recommended_attraction_ids.append(results[x][0])

recommended_attraction_ids
```

```
[40, 36, 16, 4, 9]
```

The code snippet gets location ids for the input provided for the recommender system,

These location ids are then passed to the attractions dataset to retrieve the features of the location. From the snippet below it shows the recommender has extracted user input and used this information to get data related to these ids.

```
temp = df1[df1['typeID'] == 2].sort_values("rating", ascending = False)
temp.head()
```

	attractionID	typeID	rating
2	2	2	4.5
12	12	2	4.5
26	26	2	4.5
45	45	2	4.5
48	48	2	4.5

```
history_attraction_ids = temp['attractionID']
user_history = attractions[attractions['location_id'].isin(history_attraction_ids)]
```

```
user_history[:n]
```

	location_id	location_name	location_type	ratings	no. of rating	type_id
1	1	Mount Royal Park	Parks	3.0	10,493	2
2	2	Stanley Park	Parks	4.5	28,722	2
12	12	Peller Estates Winery	Wineries & Vineyards	4.5	2,789	2
19	19	The Calgary Zoo	Nature & Wildlife Areas Zoos	3.5	4,461	2
26	26	Springbank Park	Nature & Wildlife Areas Parks	4.5	466	2

The code and results above gives the user history from the given attractionID and location ids provided by the user.

5.5. Chatbot integration of the Recommender system

The chatbot was implemented using dialogFlow based of main four intents that provide different type of recommendations to the user including:

- Providing three similar recommendations to previously visited place by the user.
- Providing the top two recommendations for the best time chosen by the user.
- Providing the top two recommendations for the province chosen by the user.
- Providing the top two recommendations for attraction type chosen by the user.

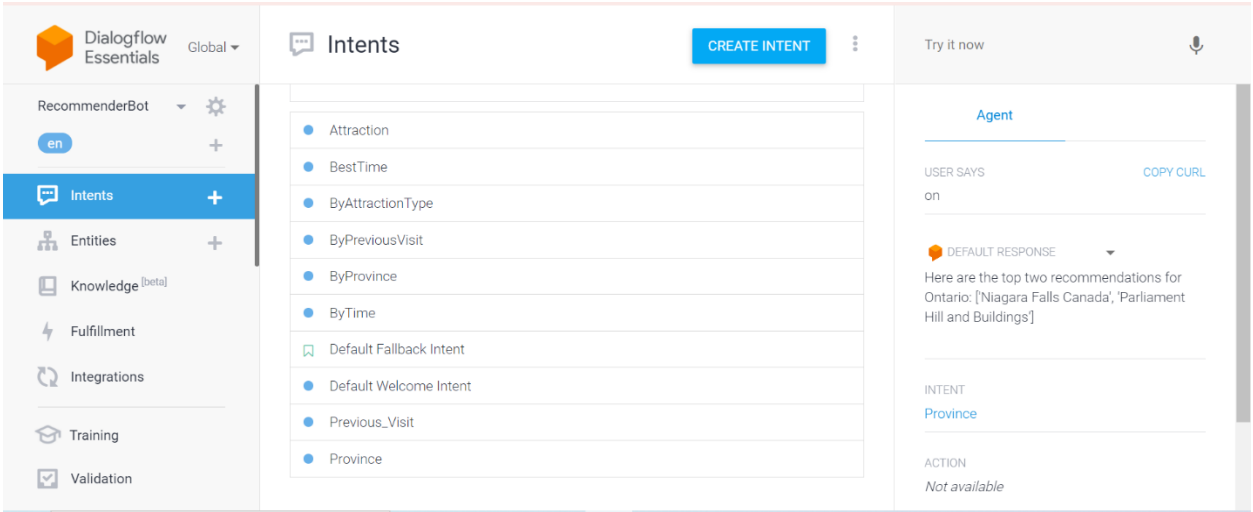


Figure 30: Intents for the chatbot.

To link the chatbot by the recommender system to get dynamic responses, we used flask to create routes to receive and send information from/to chatbot using ngrok as a tunnel between them over a local host as shown in diagram below.

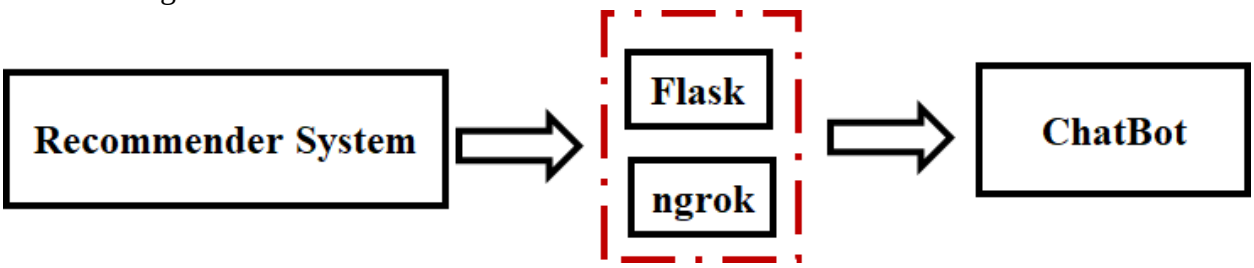


Figure 31: Integerating the chatbot with reommnder system.

6. Error analysis

In addition to the error analysis discussed above, a quick recap on the algorithms is necessary. In this project, we used the following content based filtering models

- based on the type of location tourists will like to visit
- Based on the province of location tourist will like to visit
- Based on the best time to visit the locations of interest
- Based on the user profile using multiple combined features
- based on k-means clustering algorithm for unsupervised learning
- Collaborative filtering model was used using multiple libraries in the surprise library and using multiple classification algorithms in sc-kit learn-library for supervised learning.

From the content based filtering based on multiple sources the model was unable to predict when the string entered is not a match i.e. different typed string or misspelled words threw the machine off. In more robust implementation a proper enhancement of this algorithm using fuzzywuzzy will be used for accurate prediction.

As described earlier, the SVM has poor performance when user_ids are considered as labels with 22% accuracy. Considering this difference, more emphasis was laid on location ID instead because this gave an accuracy of 44%.

Although, the results are still not very high but this can be attributed to the smaller number of rows in our dataset and a thousands of rows for the training set is required as well.
In addition, with collaborative filtering using surprise library with the various algorithm variations fully packed in the surprise algorithm (SVD, SVDpp, SlopOne, NMF etc) comparing all the models shows KNNwithZScore gives the best performance with RSME.
To conclude on the analysis used for the system, results were evaluated based on ratings using grid search cross validation to compute the accuracy metrics for an algorithm on various parameter combinations for K, K_min and min_support. This produced an RSME score of **0.969**

7. Results

We analyzed results and checked accuracy for multiple models. We also did error analysis on these and after considering all the factors, we concluded that Content Based Model using Cosine Similarity as our champion model as it works best for our recommender system.
After choosing the champion model ,we designed our chatbot system using this model and results achieved for the same are satisfactory.

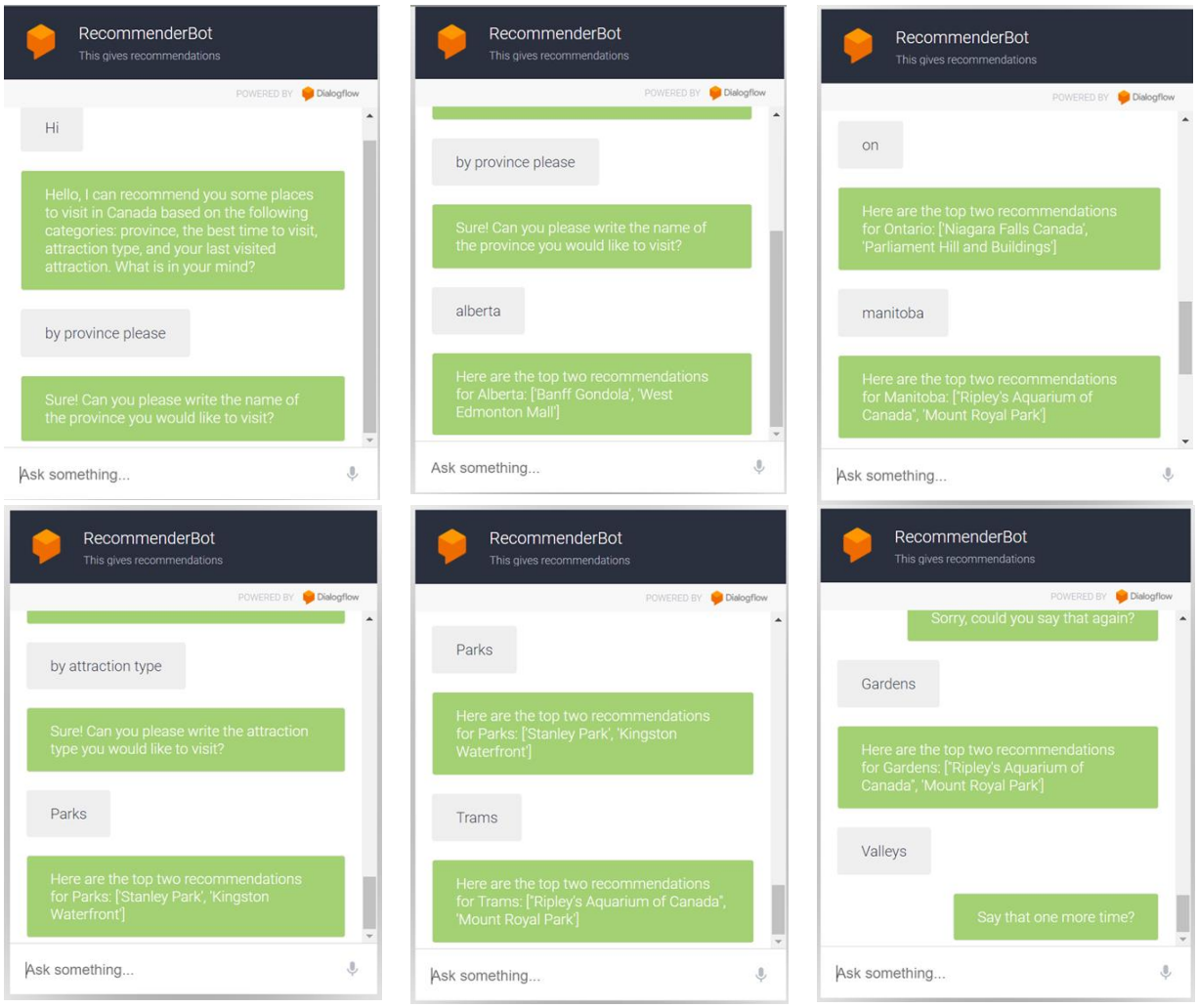


Figure 32: Shows some screenshots for conversations with chatbot

8. Insights

1. There are numerous ways to create recommender systems which performs well. The champion model depends on the requirements of the developer.
2. We analyzed the results and considered the error analysis and concluded that Content Based Model using Cosine similarity is our champion model.
3. After choosing the champion model ,we designed our chatbot system using this model and results achieved for the same are satisfactory.
4. Implementation of Chatbot involved some deep research work and new techniques and therefore proved to be very challenging.
5. Web scraping proved to be beneficial to get the required dataset from TripAdvisor.
6. Classification algorithms performs poorly due to inadequate data.
7. The dataset scrapped from the website further needed some improvisations to make it ideal for our use case.

9. Conclusion

Recommendation systems are powerful technology tool that helps to produce/ generate useful information from user database. This information can used in many ways like to recommend new items/content to the user or keep track of user activities though this dataset information and based on that make the business model successful. Through this project we tried to enhance our knowledge on the internal functioning of recommender systems and how they can be implemented using python.

10. READme

Please see attached file.

11. References

1. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/
2. https://www.pluralsight.com/browse, For finding the original number of categories as '8' for all courses.
3. https://stats.stackexchange.com/questions/155880/text-mining-how-to-cluster-texts-e-g-news-articles-with-artificial-intellige , A possible solution improvement from this thread. First try the svd approach[Currently, using similar to this approach only] and then doc2vec approach.
4. https://stackoverflow.com/questions/19197715/scikit-learn-k-means-elbow-criterion, Elbow method to find original value of 'k'.
5. https://towardsdatascience.com/build-your-own-clustering-based-recommendation-engine-in-15-minutes-bdddd591d394
6. https://pythonprogramminglanguage.com/kmeans-text-clustering/, Key resource in-use for clustering.
7. https://github.com/sachinnpraburaj/Intelligent-Travel-Recommendation-System
8. https://pub.towardsai.net/recommendation-system-in-depth-tutorial-with-python-for-netflix-using-collaborative-filtering-533ff8a0e444
9. https://towardsdatascience.com/how-to-build-a-flexible-movie-recommender-chatbot-in-python-f111da4039c1
10. https://www.analyticsvidhya.com/blog/2020/08/recommendation-system-k-nearest-neighbors/
11. https://github.com/vivdalal/movie-recommender-system
12. https://github.com/shr1911/Tourism-Recommendation
13. https://github.com/maitreytalware/Recommender-system-for-tourists
14. https://github.com/somyamakkad97/Tourist-Recommender-System
15. https://github.com/srirambaskaran/hybrid-movie-recommender-system

16. https://github.com/jkhlr/hybrid-recommender-system
17. https://www.kaggle.com/robottums/hybrid-recommender-systems-with-surprise
18. https://www.kaggle.com/niyamatalmass/lightfm-hybrid-recommendation-system
19. https://github.com/sinchana-eshwar/Movie-Recommender-System
20. https://github.com/amanjeetsahu/Recommender-Systems-Using-Python
21. https://github.com/prakruti-joshi/Movie-Recommendation-System
22. Hybrid model (content based + popularity based + item-item CF + svd)
23. https://github.com/SebastianRokholt/Hybrid-Recommender-System
24. https://github.com/Hasibul-Islam/Hybrid-Recommendation-System
25. https://github.com/amolratnaparkhe/Foodfinder-Recommendation-System
26. https://medium.com/fnplus/evaluating-recommender-systems-with-python-code-ae0c370c90be
27. https://www.researchgate.net/figure/Content-based-filtering-vs-Collaborative-filtering-Source_fig5_323726564
28. https://www.artificialintelligence21.tech/2020/07/collaborative-filtering-and-content.html