City Recommendation in the USA using Yahoo Flickr Creative Commons 100M Dataset

by Anant Jain, Ahmet Salih Gundogdu

DS 5500 Fall 2018 --- Prof. Cody Dunne, Northeastern University

Motivation:

While the advancement of transportation industry in the last hundred years enabled more travelling options to the people, it wasn't until just half a decade ago, with the emergence of social media platforms, shared economy and GPS equipped mobile devices, that these digital footprints of human mobility became public in a real-time. In this new world of shared economy, travelling cost per person has reduced due to which more and more people are moving from one region to another leaving behind their digital trails on social media sites. Although it is still not clear how living online will affect us in future, we can be sure for one thing - the huge digital
data produced by people all around the world represents almost an inexhaustible resource that can be used in different scientific studies.

The geo-location part of travel check-ins allows researchers to concentrate their focus on regions like populated cities. More than half of the population of the entire world resides in them. However, these cities aren't studied just because a lot of people live in them, but also because they have a rich history of development and growth, which lures many travelers and tourists from all over the world.

In this era of social networking sites, people's behavior online could help us have an idea about their tastes and preferences. Social check-ins, which store geo-tagged information about users, can be utilized by researchers and businesses to investigate these preferences and improve their services. In this project, we utilize the travel check-in data and use data-based visualizations to explore, assess and evaluate multiple SVD algorithms for the purposes of identifying anomalies, generating trust and providing the best recommendation for cities to visit in the USA.

Data:

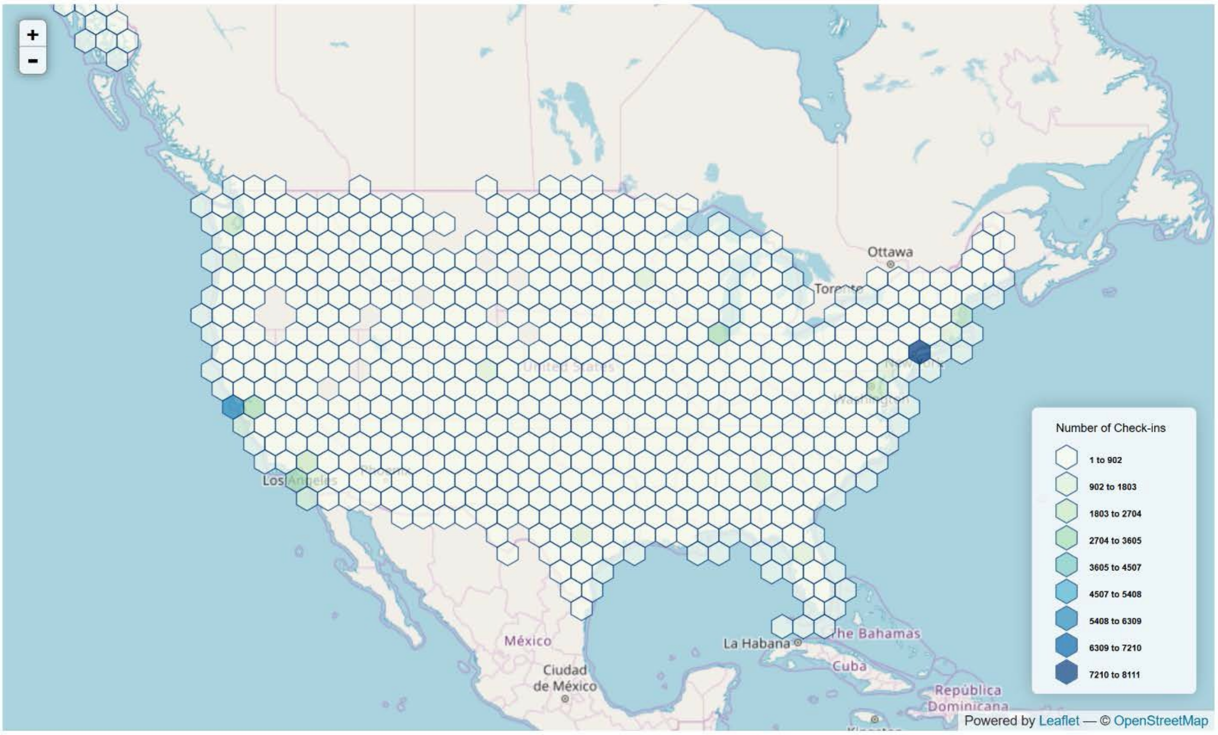
We are using Yahoo Flickr Creative Commons (YFCC100M) dataset which is known to be one of the largest assemblages of multimedia check-ins ever created. It is publicly hosted on AWS and was released under the Yahoo Web-Scope program. It comprises of hundred million media objects dating between 2004 and 2014. Each line contains the metadata for one media object. The metadata is cut and tab-separated into following field names: media identifier, user identifier, date taken, date uploaded, capture device, location (longitude & latitude, if available), accuracy and Creative Commons license it was published under. In addition, metadata fields like object title, machine tags, user tags, media description and a link from where the media object can be downloaded are listed.

We pruned the data first by eliminating unwanted columns in order to make it workable with limited RAM; and then by omitting records that weren't geo-tagged (i.e. more than 50%). Since we are focusing the study only on the cities in the US, we further filtered the data to records checked in the US. To do so, an expansion of YFCC100M containing the reverse geocode information of all records was utilized. After both pruning and merging, we were left with about 16 million records and the following set of columns.

Column	Description	Data Type
pid	Unique media identifier	categorical
user_nickname	User identifier	categorical
date_taken	Date the media object was created	ordinal
longitude	Longitude of the location the media object was checked at.	quantitative
latitude	Latitude of the location the media object was checked at.	quantitative
url	Link from where the media object can be downloaded	categorical
town	City the media object was checked in, extracted from address column	categorical
state	State the media object was checked in, extracted from address column	categorical

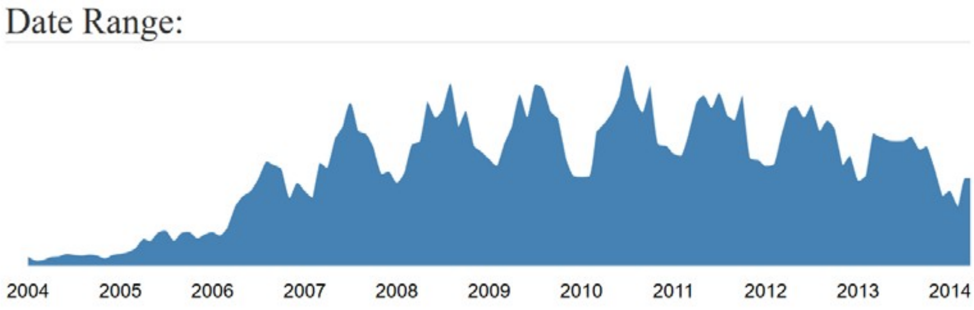
Data Analysis:

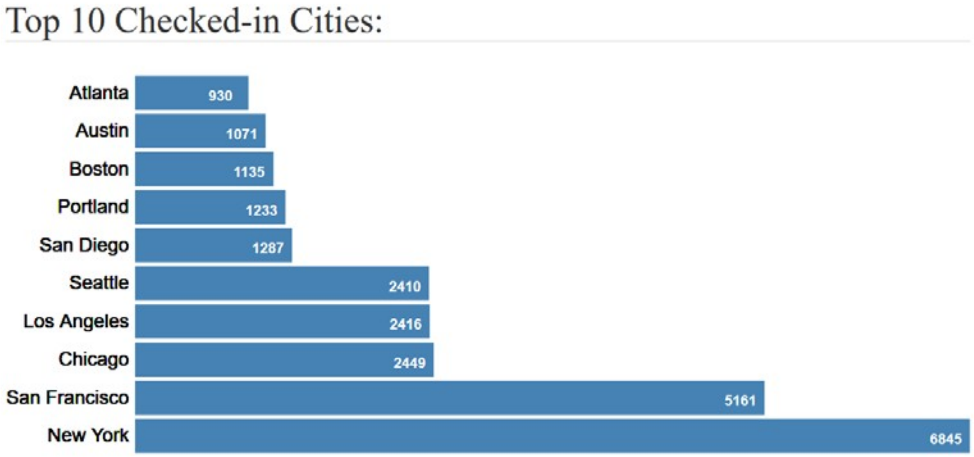
Visualization 1: Check-ins hex-binned by location on Map



The above visualization shows a hex-binned map of a sample of data. Hex-bins are used to encode location of check-ins and color is used to encode number of check-ins in that area. We see most of the check-in concentration is around cities like New York and San Francisco. The middle of the US has the least number of check-ins. It seems most of the popular cities which users like to travel to are situated on the coasts of the US.

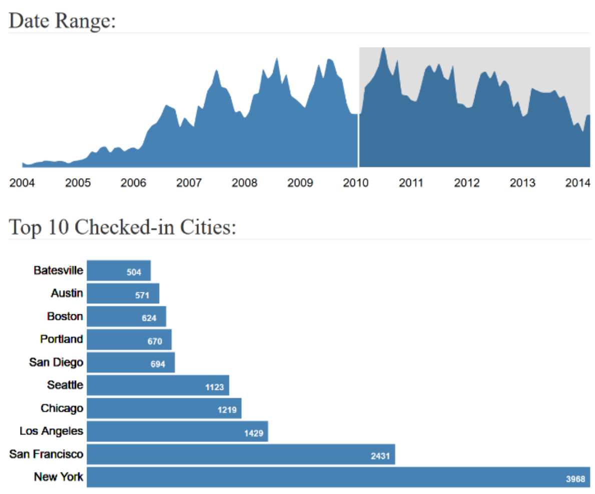
Visualization 2: Top-10 Cities in Specified Date Range

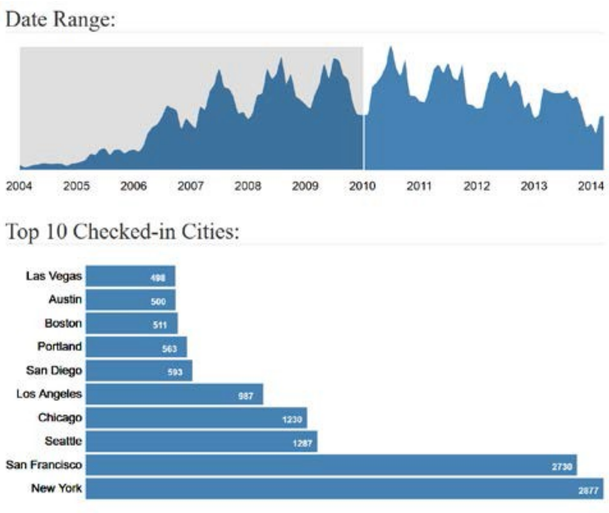




The above visualization shows an area chart which encodes date data on the x axis and number of check-ins on the y axis; accompanied by a bar chart which encodes the top-10 cities on the y-axis and number of check-ins on the x-axis in the specified date range. We clearly see the leading cities across the US.

Fun-fact: As we can see in the chart below, before 2010, the difference in the number of check-ins between that of New York and San Francisco isn't that much. But, after 2010, We see New York almost doubled it's count of check-ins and became the sole most popular city in the US.





Task Analysis:

Priority	"Domain" Task	Analytic Task	Search Task	Analyze Task
3	Examining and evaluating the model performance of the recommended places against the given user’s travel history	Compare	Locate	Present
2	Generate a ranked list of recommendations	Sort	Explore	Present
1	Visualize different models and hyperparameters for assessment of the best set of modeling parameters to use.	Compare	Explore	Discover
4	Exploratory Data Analysis	Compare	Explore	Discover

Our visualizations will be primarily developed for "discover" (i.e., exploratory visualizations), "present" (i.e., communicative visualizations) consumptions. These domain tasks will enable our potential users to have more understanding and flexibility of choosing different models for the location recommendation task. More on why we chose the specified tasks are as follows:

1. Priority #3: As the model will predict the given user's tendencies as a number, we will be sorting the top 10 locations rather than calculating the proximity to the real location. Here we will evaluate our model performance using Precision@10 metric which is looking at the user's top ten location preference and predicted top 10 recommendations.
2. Priority #2: Physically demonstrate and deliver the models’ outputs to the user.
3. Priority #1: Enable user to choose their modeling approach (at least 2 different SVD adaptations) and tune their parameters and evaluation metric.
4. Priority #4: Get an idea about the data on which all the analysis is based on.

The users of the end product will be researchers and machine learning engineers who are interested in recommendation systems, but the UI is going to be designed in such a way that a normal traveler who want some good recommendations could also make use of the tool.

Model Description:

We give the user the freedom to assess and visualize different adaptations of SVD on his/her selection of hyperparameters.

We have the following options for Hyperparameter Testing and Model Selection:

Preprocessing:

1. Numeric: Counts the number of check-ins for each city per user and creates a numeric label in domain [0, inf).
2. Binary: Labels the user-location pair as 1 if user has checked-in in that city, else 0.

Models:

1. SVD_explicit: Use Singular Value Decomposition to extract user and location features and approximate the missing user check-ins by matrix factorization.
2. SVD_implicit: Use Singular Value Decomposition to extract user and location features and then apply Alternating Least Squares method to get an optimal fit for missing points.

Python libraries: **Scipy**, Scikit-learn, Pandas, Numpy, **Implicit** (Bold texts represents the packages used for building ML models)

Latent Dimensions: Denotes the number of dimensions/features to extract for each user and location.

Metric:

1. Precision-Train Set: Use Precision@10 metric which looks at the user's top 10 location preferences and compare it with the predicted top 10 recommendations for the Train Set.
2. Recall-Train Set: Use Recall as metric to assess the user's top 10 location preferences and compare it with the predicted top 10 recommendations for Train Set.
3. Precision-Validation Set: Use Precision@10 metric which looks at the user's top 10 location preferences and compare it with the predicted top 10 recommendations for the Validation Set.
4. Recall-Validation Set: Use Recall as metric to assess the user's top 10 location preferences and compare it with the predicted top 10 recommendations for the Validation Set.

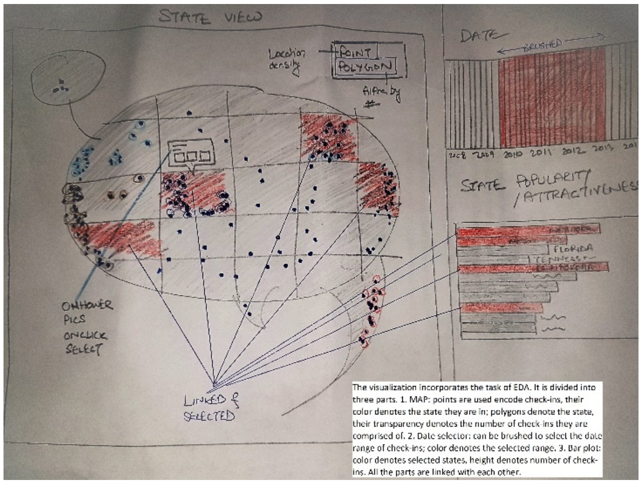
Design Process:

We started with drawing up some preliminary sketches based on our tasks. As it can be seen we were planning to have point and polygon layers on the EDA map initially. But we soon realized how over plotted it is going to look giver our dataset size. Polygon layers were supposed to the fix for that but they didn’t help much either if the user changed the level of the map. We experimented and researched on what would be the best encoding for the job and we settled with hex-bin layers.

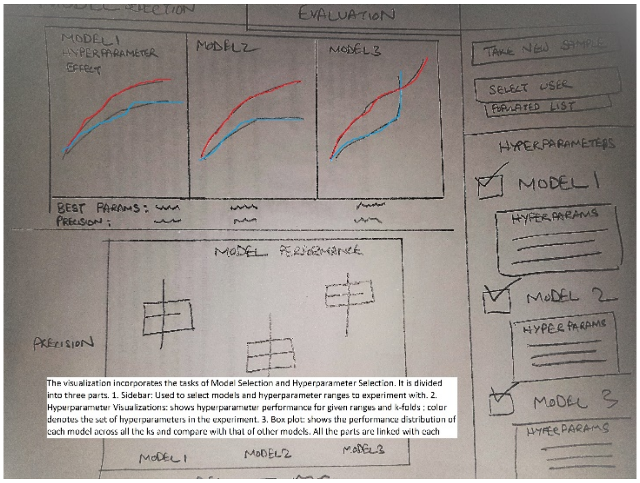
Hyperparameter Testing was implemented exactly ass planned at the initial stage. But, for Model Selection, as per the feedback we got from the instructors, we decided to change from box-plots to violin plots.

The biggest changes were made to the Evaluation and Results visualizations. We decided to drop Feature importance graph and integrate everything else on one big map. This resulted in a much cleaner UI design.

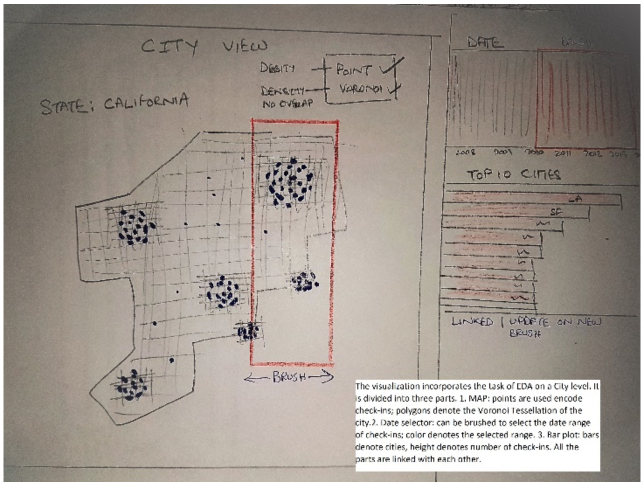
Preliminary Sketches:



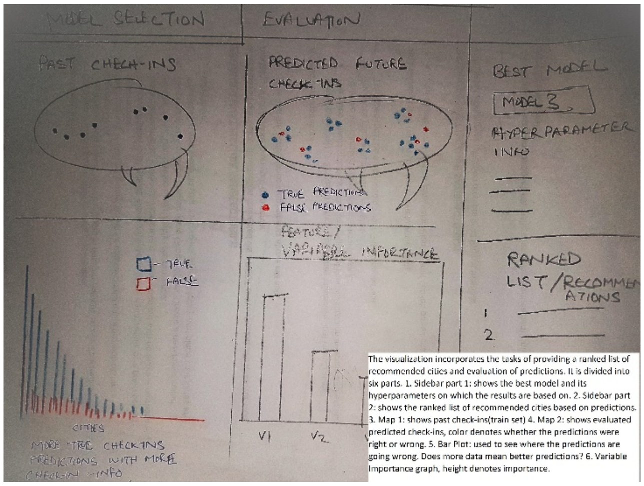
The visualization incorporates the task of EDA. It is divided into three parts. 1. MAP: points are used encode check-ins, their color denotes the state they are in; polygons denote the state, their transparency denotes the number of check-ins they are comprised of. 2. Date selector: can be brushed to select the date range of check-ins; color denotes the selected range. 3. Bar plot: color denotes selected states, height denotes number of check-ins. All the parts are linked with each other.



The visualization incorporates the tasks of Model Selection and Hyperparameter Selection. It is divided into three parts. 1. Sidebar: Used to select models and hyperparameter ranges to experiment with. 2. Hyperparameter Visualizations: shows hyperparameter performance for given ranges and k-folds ; color denotes the set of hyperparameters in the experiment. 3. Box plot: shows the performance distribution of each model across all the ks and compare with that of other models. All the parts are linked with each

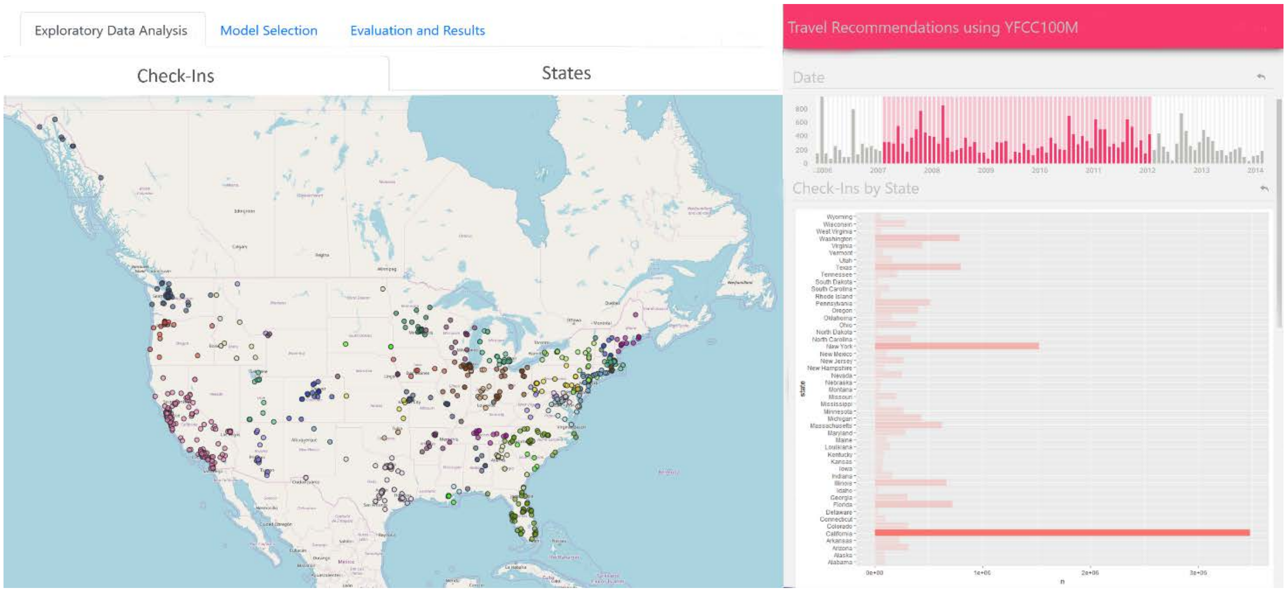


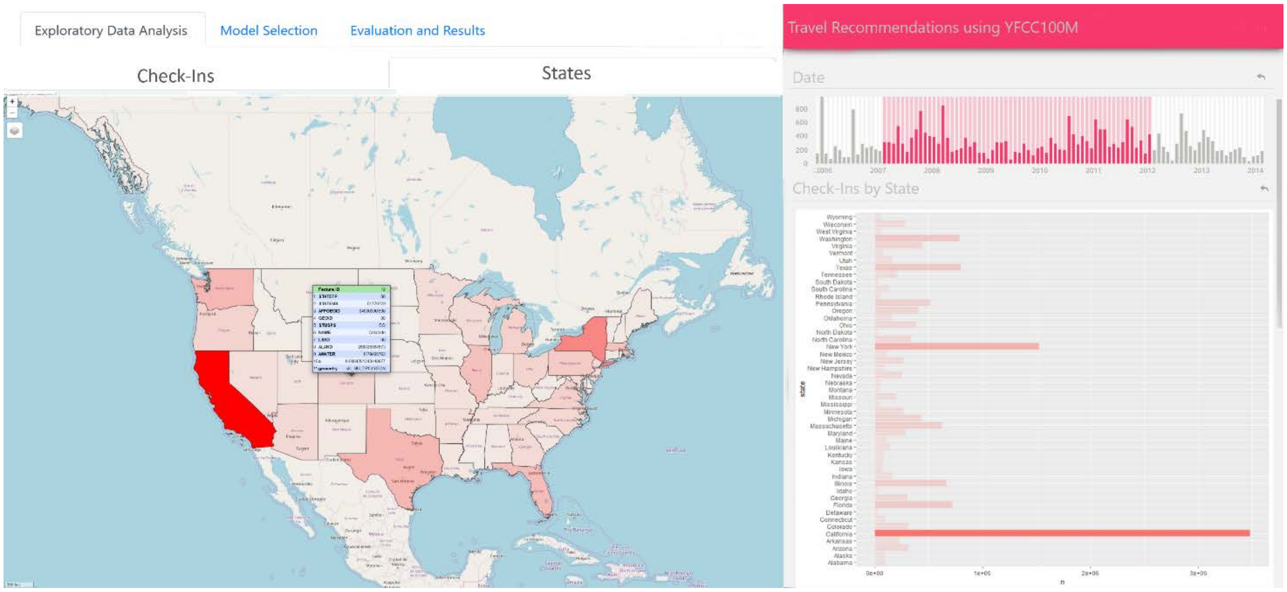
The visualization incorporates the task of EDA on a City level. It is divided into three parts. 1. MAP: points are used encode check-ins; polygons denote the Voronoi Tessellation of the city.2. Date selector: can be brushed to select the date range of check-ins; color denotes the selected range. 3. Bar plot: bars denote cities, height denotes number of check-ins. All the parts are linked with each other.

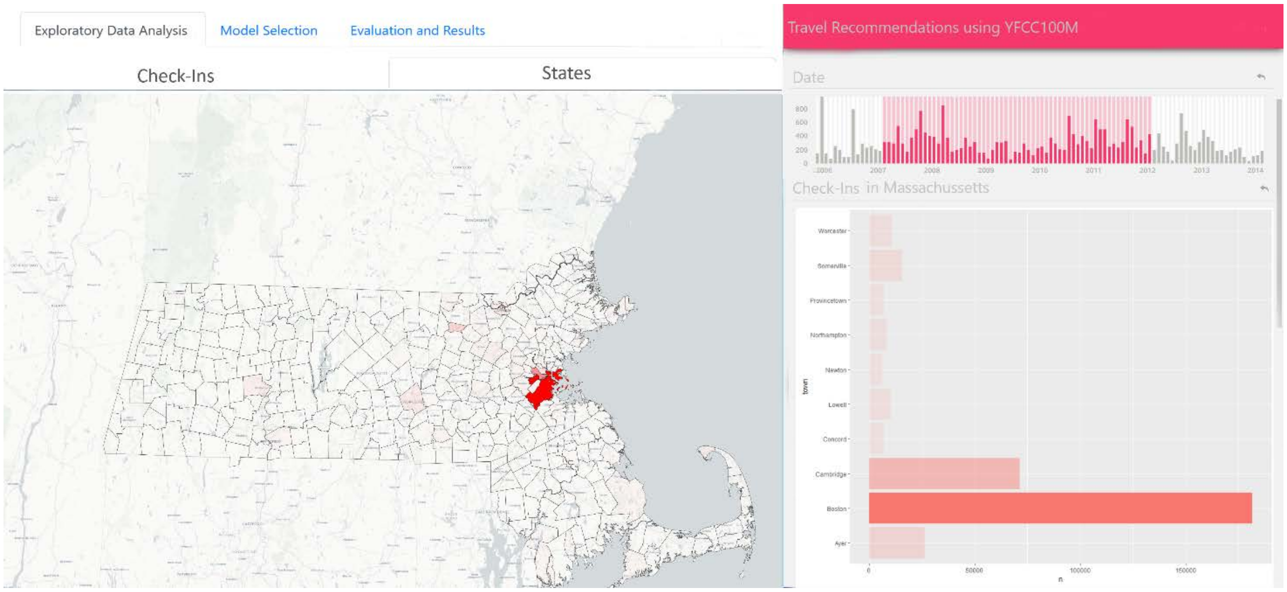

MODEL SELECTION
EVALUATION
PAST CHECK-INS
PREDICTED FUTURE CHECK-INS
TRUE PREDICTIONS
FALSE PREDICTIONS
BEST MODEL
MODEL 3
HYPERPARAMETER INFO
TRUE
FALSE
FEATURE/VARIABLE IMPORTANCE
V1
V2
CITIES
MORE THE CHECKINS PREDICTIONS WITH MORE ACCURATE INFO
RANKED LIST/RECOMMENDATIONS
1
2

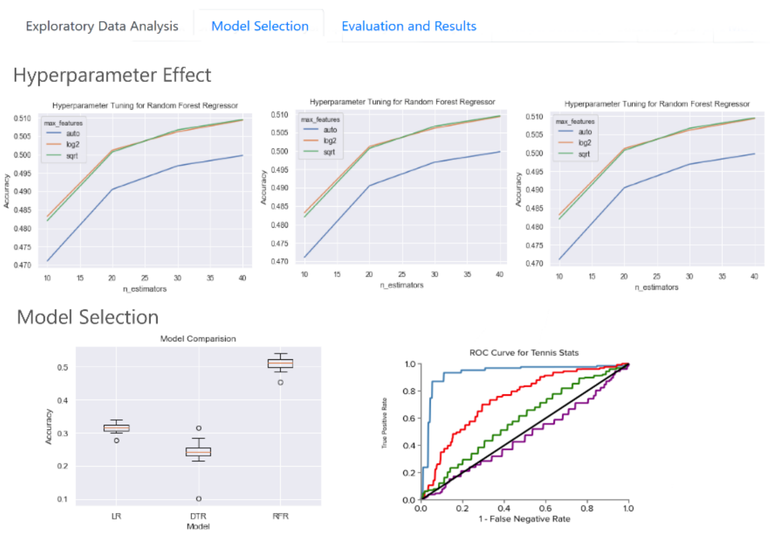
The visualization incorporates the tasks of providing a ranked list of recommended cities and evaluation of predictions. It is divided into six parts. 1. Sidebar part 1: shows the best model and its hyperparameters on which the results are based on. 2. Sidebar part 2: shows the ranked list of recommended cities based on predictions. 3. Map 1: shows past check-ins(train set) 4. Map 2: shows evaluated predicted check-ins, color denotes whether the predictions were right or wrong. 5. Bar Plot: used to see where the predictions are going wrong. Does more data mean better predictions? 6. Variable Importance graph, height denotes importance.

Digital Sketches:



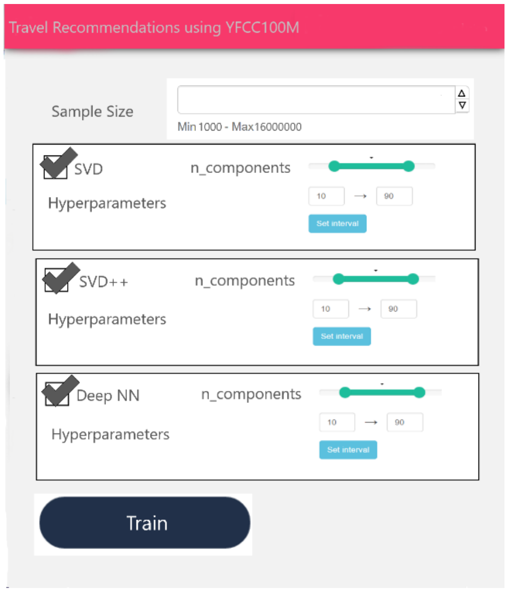






Hyperparameter Effect

Model Selection


Travel Recommendations using YFCC100M

Sample Size

Min 1000 - Max16000000

☑ SVD n_components 10 → 90

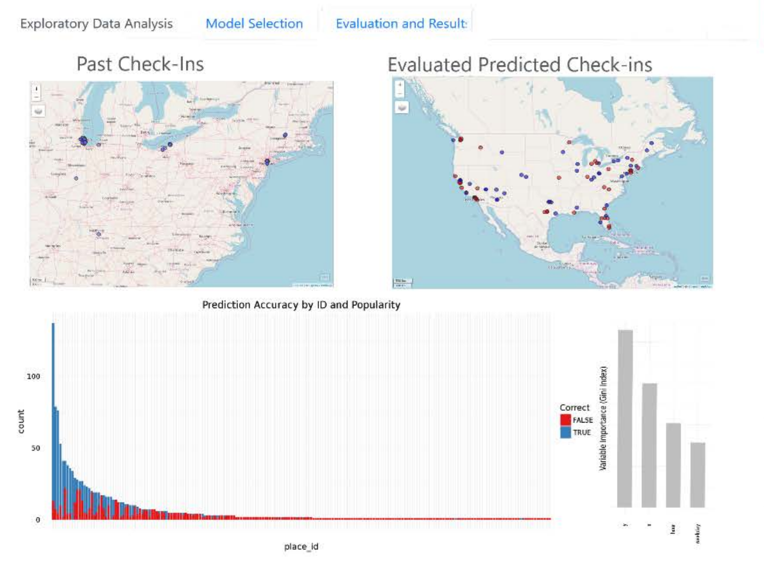
Hyperparameters

☑ SVD++ n_components 10 → 90

Hyperparameters

☑ Deep NN n_components 10 → 90

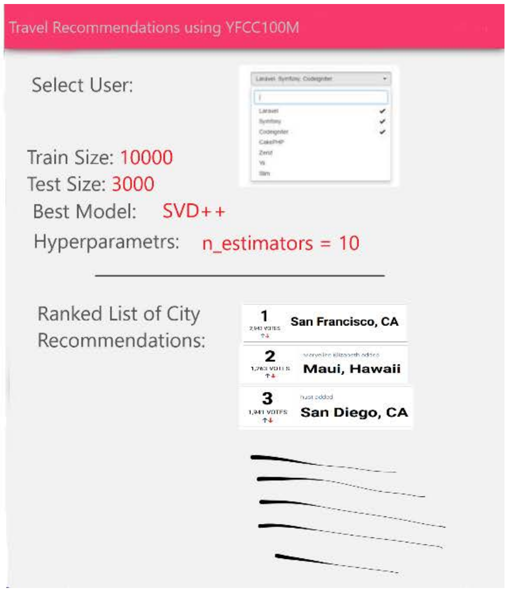
Hyperparameters


Exploratory Data Analysis
Model Selection
Evaluation and Result:

Past Check-Ins

Evaluated Predicted Check-ins

Prediction Accuracy by ID and Popularity
count
100
50
0
place_id
Correct
FALSE
TRUE
Variable Importance (Gini Index)
y
x
hour
weekday


Travel Recommendations using YFCC100M

Select User:

Train Size: 10000

Test Size: 3000

Best Model: SVD++

Hyperparametrs: n_estimators = 10

Ranked List of City Recommendations:

1 2,940 VOTES San Francisco, CA

2 1,263 VOTES Maui, Hawaii

3 1,941 VOTES San Diego, CA

Final Visualization:

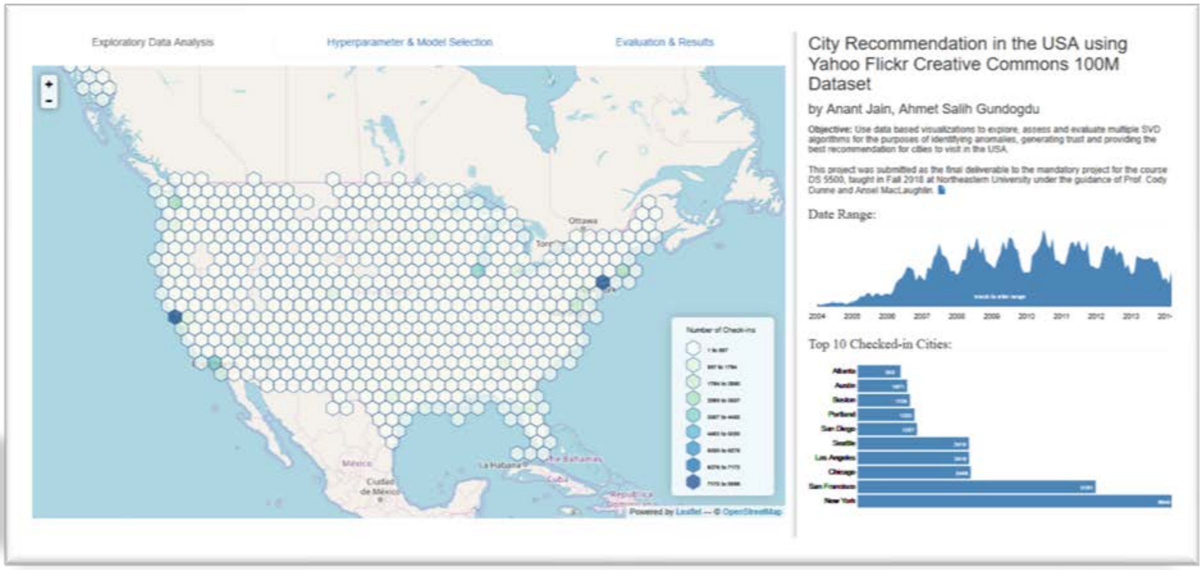
We decided to divide our final Visualization into three sections:

1. Exploratory Data Analysis
2. Hyperparameter Testing & Model Selection
3. Evaluation and Results

Each section took care of at least one task in our tasks table. All the three section are comprised in a Bootstrap UI with a main panel and a side-bar panel.

We'll go by each section and discuss its purpose, design justification and packages utilized for coding.

Exploratory Data Analysis:



City Recommendation in the USA using Yahoo Flickr Creative Commons 100M Dataset

by Anant Jain, Ahmet Salih Gundogdu

Objective: Use data based visualizations to explore, assess and evaluate multiple SVD algorithms for the purposes of identifying anomalies, generating trust and providing the best recommendation for cities to visit in the USA.

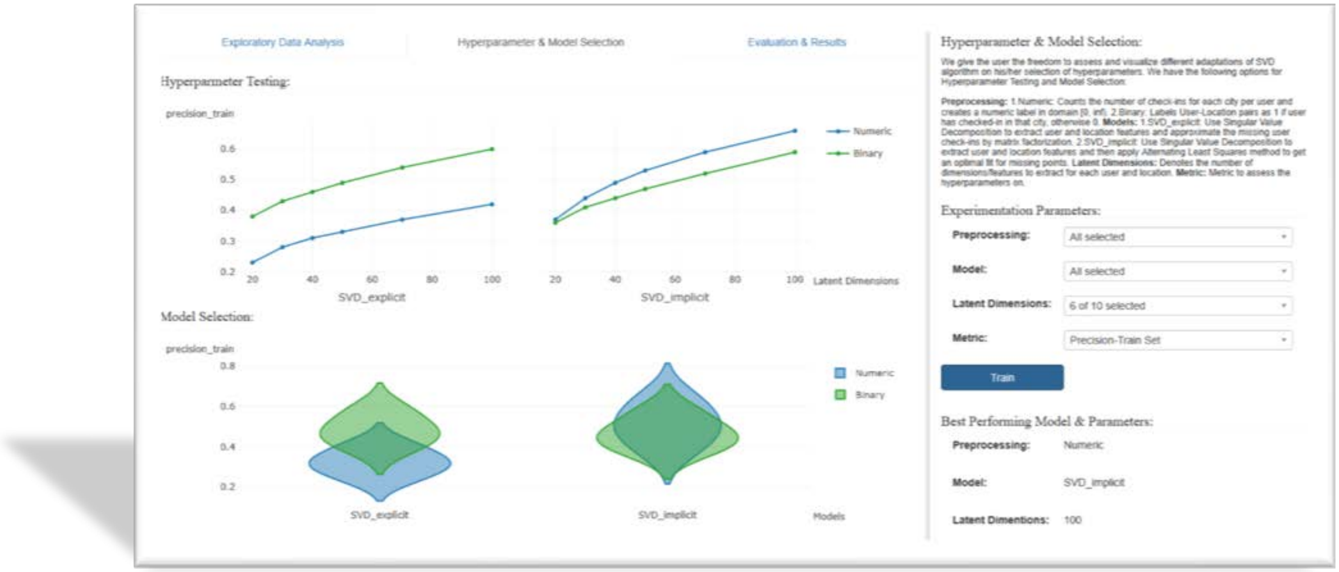
This project was submitted as the final deliverable to the mandatory project for the course DS 5500, taught in Fall 2018 at Northeastern University under the guidance of Prof. Cody Dunne and Ansel MacLaughlin.

Exploratory Data Analysis is the first tab on the UI.

The main panel shows an interactive leaflet map with hex-binned layers. The position of the hex-bins ideally encodes location of the aggregated check-ins at each zoom level of the map. The color of the hex-bins encodes the number of check-ins in that aggregated region. A Green-Blue color scale from ColorBrewer is used. A user can investigate the number of check-ins contained in each hexagon by hovering the mouse pointer over the hexagon (this is going to show a tipsy tooltip).

The side-bar panel consists of an area chart and a bar chart made using D3. The area chart encodes the number of check-ins on the y-axis and check-in dates on the x-axis. This chart can be brushed and is linked to the map and the bar chart. Meaning, a user can filter the data shown by the visualizations on the page by date just by brushing the area chart. The bar chart encodes the top 10 checked-in cities/towns on the y-axis and the number of check-ins on the x-axis. As it is linked to the area chart, it updates accordingly with respect to the brushed date-range.

Hyperparameter Testing & Model Selection



Hyperparameter & Model Selection:

We give the user the freedom to assess and visualize different adaptations of SVD algorithm on his/her selection of hyperparameters. We have the following options for Hyperparameter Testing and Model Selection:

Preprocessing: 1.Numeric: Counts the number of check-ins for each city per user and creates a numeric label in domain [0, inf). 2.Binary: Labels User-Location pairs as 1 if user has checked-in in that city, otherwise 0. **Models:** 1.SVD_explicit: Use Singular Value Decomposition to extract user and location features and approximate the missing user check-ins by matrix factorization. 2.SVD_implicit: Use Singular Value Decomposition to extract user and location features and then apply Alternating Least Squares method to get an optimal fit for missing points. **Latent Dimensions:** Denotes the number of dimensions/features to extract for each user and location. **Metric:** Metric to assess the hyperparameters on.

Experimentation Parameters:

Preprocessing: All selected

Model: All selected

Latent Dimensions: 6 of 10 selected

Metric: Precision-Train Set

Best Performing Model & Parameters:

Preprocessing: Numeric

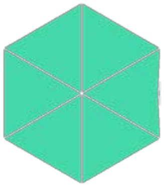
Model: SVD_implicit

Latent Dimentions: 100

Hyperparameter Testing & Model Selection is the second tab on the UI.

The main panel is divided into two parts. First shows a multi-columned line chart to explore hyperparameter test results. The y-axis encodes the metric to evaluate the hyperparameters on and the x-axis encodes the latent dimensions of user's choice. The column encodes the type of model for which the user wants to view the test results on. Second shows a violin plot encoding the metric on y-axis and model on x-axis. Area inside the violin denotes the distribution of metric for different latent dimensions chosen by the user. The color for both the chart encodes the types of preprocessing the user has chosen. All the visualizations in this section were made using plotly.

The side-bar panel has bootstrap and multi-select input elements so that the user can input his/her's set of hyperparameters and models to explore and assess. Once the user hits the train button, the page makes a request to the flask server to get the data for the visualizations and show the best set of hyperparameters for entered inputs in the sidebar. The page fires a request to the get the data for default parameters when the user first switches to the tab.



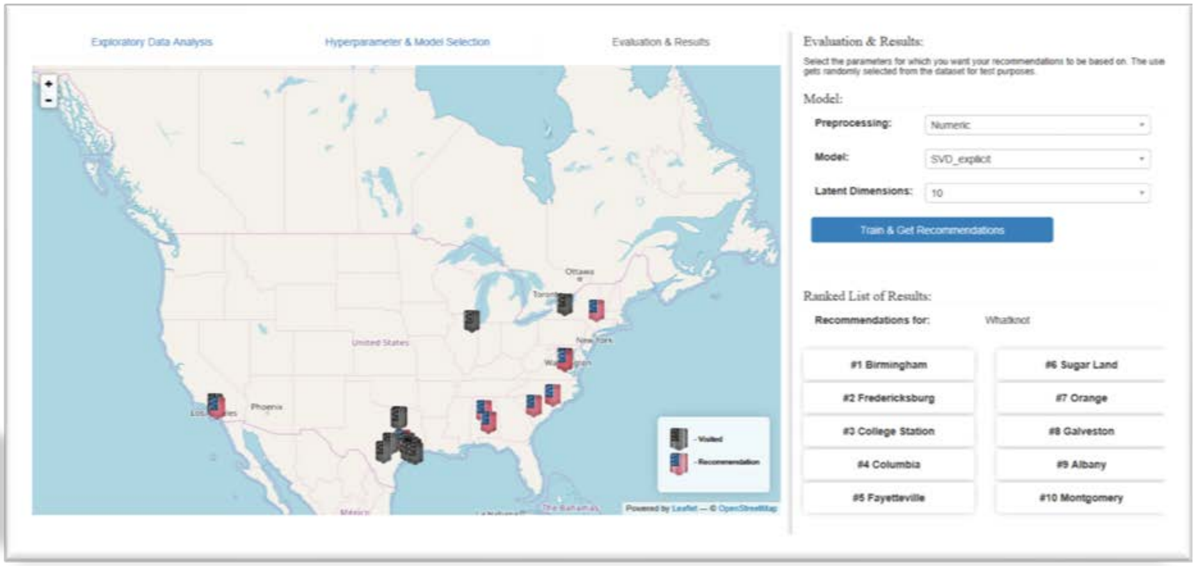
It might take some time for the page to receive the data from the server therefore a user is going to see the UX gif in the main panel. A loading hexagonal gif means the page has fired up a query to the server and is fetching the data for the visualization.

Evaluation and Results

Lastly, we have Evaluation and Results on the third tab of the UI.

The main panel shows an interactive leaflet map with animated uncopyrighted icons to encode visited towns (bnw flag) and recommended towns (colored flag) with details-on-demand hover functionality.

The side-bar panel accommodates the input elements for the final set of parameters the user wants his recommendations to be based on and a ranked list of results/recommendations for a random user which automatically gets selected from the database.



Evaluation & Results:

Select the parameters for which you want your recommendations to be based on. The use gets randomly selected from the dataset for test purposes.

Model:

Preprocessing: Numeric

Model: SVD_explicit

Latent Dimensions: 10

Ranked List of Results:

Recommendations for: Whatknot

#1 Birmingham	#6 Sugar Land
#2 Fredericksburg	#7 Orange
#3 College Station	#8 Galveston
#4 Columbia	#9 Albany
#5 Fayetteville	#10 Montgomery

Conclusion:

The overall goal of this project was to utilize the travel check-in data and use data-based visualizations to explore, assess and evaluate multiple SVD algorithms for the purposes of identifying anomalies, generating trust and providing the best recommendation for cities to visit in the US. Although using Singular Value Decomposition for Recommendation Systems is a fairly common approach; using visualizations to tune the hyperparameters of these models in real time and making the user a part of the machine learning process helps him/her to develop confidence in the results. While visualizations can be appealing and persuasive, it is important to have the proper visual encodings to use in the visualizations otherwise instead of making the user build trust in the results, they could easily push the user away.

Even though this project was a simple implementation of recommendation systems using Singular Value Decomposition, it can be scaled to integrate much complex deep-learning models designed for the tasks of collaborative filtering (i.e. Auto-Encoders). In addition to integration of complex models, distance-based evaluation techniques, where the results are evaluated on the basis of the gap between predicted recommendations and ground truth, are also suitable for future work.

Evaluation Plan:

In order to implement this design in real-time and real-world, an evaluation plan to measure the functionality, effectiveness, efficiency, usability and usefulness of the town recommendation system is required.

First, offline evaluations on simulated data, using streaming of existing data, can be run at different time splits for training and test sets. This will basically enable us to see the model's robustness for changing data and will help in estimating how frequently we need to retrain the model. All of these offline evaluations should require around 5%-10% of the total budget planned for evaluation plan.

After completing 1-3 iterations on the offline evaluations comes the more challenging and expensive part, which is to deal with the wild environment i.e. online testing/beta testing. About 10% of the total budget can be spent on understanding the environment and work practices for this task. We can identify target groups/companies who are interested in inferring location recommendations for their users. In this project, we can recommend relevant cities to Flickr users with precision; however, there is no evidence that the same model would also work for other big domains such as Instagram, Reddit, Foursquare etc. To test whether a model trained using YFCC100M data set can recommend relevant cities for other domains as well, we can build controlled test scenarios to evaluate the precision@10 and recall@10 values for these additional datasets from other big companies. Only then, the effectiveness of the product will become more evident and can be regularized if there is a need based on the performance of this evaluation setup. This will enable us to understand YFCC100M data's predictive power for other platforms.

Also, for evaluating the system response time, we should first invest more on code optimization and then evaluate the performance using different kinds of devices (mobile, PC, etc.) and locations all over the world which can require another 10% of the budget designated for evaluation tasks.

To understand visual reasoning, we want to quantitatively measure users' engagement with our website. To do this we can apply A/B testing [Munzer, Controlled Experiment] for N number of different visualizations and monitor variables like length of stay on the website, number of clicks and so on. 5% of the budget can be spent for this. In the end, we can choose the best k (where k<<N) number of visualizations to be evaluated qualitatively.

To qualitatively evaluate this tool, we can organize large-scale usability tests accompanied by a survey for the testers to understand the tool's limitations. We can organize this evaluation task using Amazon Mechanical Turk which is a crowdsourced Internet marketplace and enables individuals to publish and get answers for the surveys on specific tasks. This should require no more than 5% of the budget. Further, we can use web application and visualization experts to evaluate this product. This step would be more expensive than the previous one and would require about 10% of the budget, but might be really effective and useful. These two usability testing tasks should be done chronologically. After the quantitative evaluation, we can heed to the feedbacks by surveys and experts. As the MTurk survey costs less money, we can easily afford to do it multiple times.

We can iterate the whole online evaluation plan twice to meet the user expectations. This should cost almost the same amount of money as it did when done for the first time. In the end, the whole evaluation process should cost less than the planned amount set for the process.