Evento

Mugdha Wadikar
Computer Engineering Department
San Jose State University
San Jose, USA
mugdha.wadikar@sjsu.edu

Jai Vilas Chhatre
Computer Engineering Department
San Jose State University
San Jose, USA
jaivilas.chhatre@sjsu.edu

Amruta Saraf
Computer Engineering Department
San Jose State University
San Jose, USA
amruta.saraf@sjsu.edu

Shivam Waghela
Computer Engineering Department
San Jose State University
San Jose, USA
shivam.waghela@sjsu.edu

***Abstract* - Evento is a complete event hosting and recommendation application. The most unique feature of the application is the "prediction" feature, which allows users to find potential number of users interested in a particular event even before hosting the event and allows them to plan it and manage funds more efficiently.**

It also allows people to browse events that are tailored according to their interests based on past events attended and proximity of the event from their location.

I. PROBLEM STATEMENT

Build a web application that lets users host events, predict the number of users interested in a particular event before it is hosted and view personalised recommendations based on the past events they were interested in. To demonstrate the inspiration behind the problem statement, we have made a small video of our application. Here is the link for it: https://www.youtube.com/watch?v=XSQ5L0YCakY

II. PERSONAS

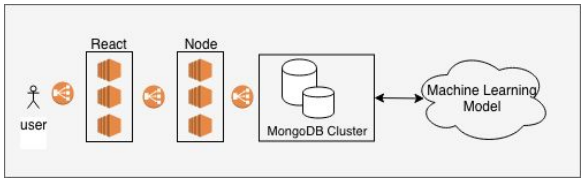
Personas for the Evento application are:

a. People who want to get an estimate of how many users will be interested if they decide to host a particular event.

b. People who want to know about events taking place around them.

c. People who want to get recommendations based on the type of events they have attended in the past and avoid going through a huge list of upcoming events, which they may not be interested in.

III. ARCHITECTURE



A. Frontend:

We used the most popular JavaScript Library - React.js to build the user interface. React.js helped us build very fast and optimized web application. As React.js only re-renders a small portion of the DOM structure that has changed, it makes the re-rendering faster and reduces the resource utilization on the client side. Along with it we used frameworks and libraries such as bootstrap and redux for state management. We hosted the React client on the AWS EC2. The deployment is done to maximize scaling and fault tolerance.

B. Backend

The backend server is a Node.js server deployed on EC2 with Auto-Scaling and Load Balancing features. The advantage of using Node.js is that it is an asynchronous JavaScript runtime engine. It offers non-blocking I/O mechanisms which prevents the main thread from getting blocked by some heavy I/O calls. This makes the server scalable and highly concurrent in processing large amounts of requests.

C. Database:

We have MongoDB database behind the server to store the flexible data. MongoDB is a highly scalable and flexible CP system. It provides high consistency and partition tolerance. The advantage of using MongoDB is that it is schemaless document-oriented NoSQL DB which will adopt to the changing requirements in the evolution of the application. The MongoDB is deployed as cluster of five nodes as an AWS EC2 instances to provide high availability and partition tolerance.

D. Machine Learning (Prediction and Recommendation)

Data:

In our prediction and recommendation system, we used the dataset of a Kaggle competition (The link of this is shared in the references section). The provided data had lots of tables. However, in this project we have worked only on the most important two features and hence we focussed only on the tables related to those features, which were:

a. users.csv
b. events.csv
c. training.csv
d. test.csv

users.csv consisted of the user information like location etc, events.csv consisted of events' information like location etc. Training.csv and test.csv had same columns like userId, eventId, interested etc.

Towards the end of our project we created our own training.csv with two engineered features in order to train our model. Feature-engineering will be explained in the following sections.

Data preprocessing:

The dataset required preprocessing as there were some inconsistent format and many

missing. We handled the preprocessing in the following way:

Missing values: we replaced the missing values with NA in the users data. In case of training data, the "interested" column (the column that served as the groundtruth) had also missing values. We removed the rows containing missing "interested" values, since we figured that they won't contribute to the "learning" part of the prediction-recommendation system.

Inconsistent location format: the users and events tables had inconsistent location format. We figured that for measuring distance between the user's and the event's location, only the city and the state is enough. Almost all of the rows in both the tables had this information so the inconsistent location no more posed a challenge for us.

For doing all the data preprocessing, we used IBM Watson Studio by creating a Data Refinery project (in the Data Engineering tab). It made all the preprocessing operations very easy to perform.

Feature Engineering:

Based on the data obtained from the Kaggle competition, there was scope for lot of features, but we narrowed in to just two features we thought were most important, i.e. proximity and relevance, for a pair of user and event (u,e). Each of the feature and the method of engineering it is as follows:

Proximity:

This feature is basically the distance between the user's location and the event's location. We figured that a user is likely to click on "interested" if the event is within the user's proximity.

To generate this feature for each pair of user and event, we have used the geopy library of python. By providing the city and state of every user and event to this library's functions, we get the distance in miles for every pair.

In the application, when the user selects either prediction/ recommendation option, user's location(which is taken from the user's profile) and event location(which is mentioned in the event's details) are provided to the python code. The python code generates the proximity feature, along with the relevance feature (described later in the report) and accesses the trained model exposed as an API(details are mentioned lated in the report) in order to generate the result

Relevance:

This feature is basically the cosine similarity between two events. In case of prediction system, when a user enters the details of the events and clicks on "predict" button, the event's description is compared with that of other past events in Evento's database and based on how many users clicked "interested" for those past events, a number is predicted for this event.

In case of recommendation system, when the user clicks on "view recommendations",

every upcoming event is compared with the past events of the user (for whom we are recommending), for which the user clicked on "interested". Only those events are displayed as result for which the cosine similarity with the past interested events is higher.

For getting the cosine similarity between two events, we have used TFIDF vectorizer of scikit-learn library. The TfidfVectorizer tokenizes the event's description, learns the vocabulary(both using "fit"operation) and inverse description frequency weightings, and we are able to encode descriptions of upcoming events(using "transform" operation). This results into a sparse array for each event description. These sparse arrays are then used to find cosine similarity between the descriptions of the two events. We have used sklearn for cosine similarity.

Training

We have used IBM Watson Studio to train our model using logistic regression analysis. We used the following services :

1. Watson™ Studio
2. IBM® Analytics for Apache Spark for IBM Cloud
3. IBM Cloud Object Storage
4. IBM Watson™ Machine Learning
5. IBM® Db2® Warehouse on Cloud

Following are the relevant screenshots from the training step using Watson Studio:

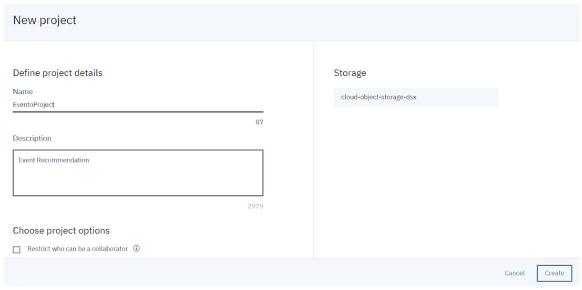


Fig. 1: creating new project in Watson Studio

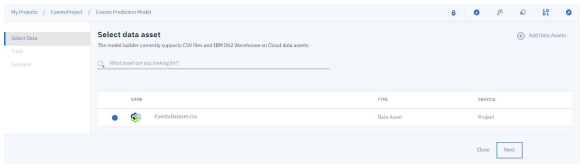
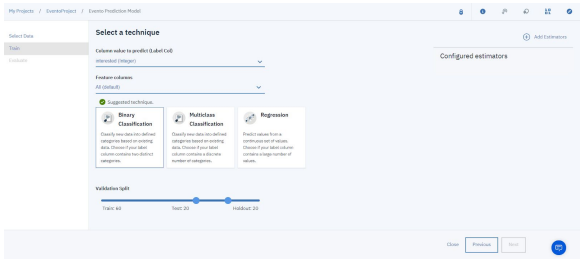


Fig. 2: add data assets. In this step we added our training.csv



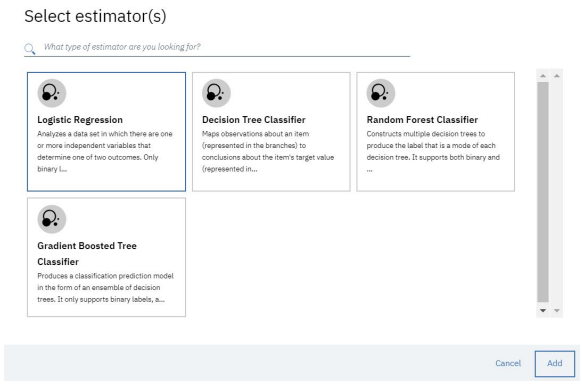


Fig. 3: select technique for machine learning. We have selected Logistic Regression

Deployment

We used IBM Watson Studio for deployment of the trained model as well. After deploying it, we were able to access our model into our web

application through the API provided by Watson studio. We followed the following steps:

Fig. 4: Create a deployment model.

After deploying the application, we were able to access the API through the service credentials(url, username and password). These service credentials were then used in a Python code to get required output from the trained model's API.

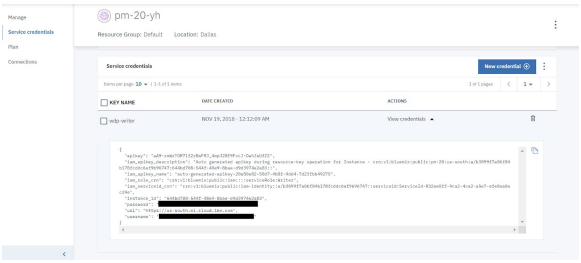
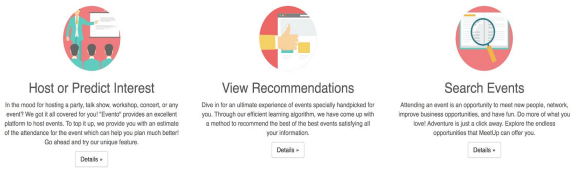


Fig. 5: Service credential of trained model API

IV. FEATURES

The application provides three main features to users -



Event Hosting and Prediction

Users can host events. They can also enter event details and get prediction of number of users that will be interested if such an event is hosted.

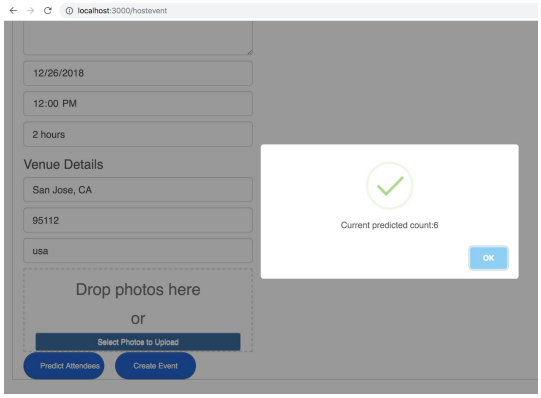


Fig. 6: Prediction of number of interested users

Personalised recommendations:

Based on users' past history of events, for which they clicked "interested", the users can view recommendations for similar upcoming events.

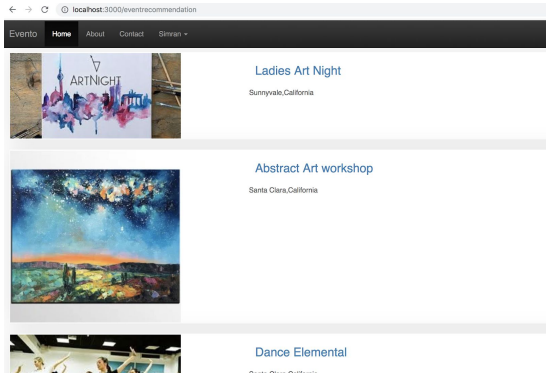


Fig. 7: Recommendation results

Browsing of event

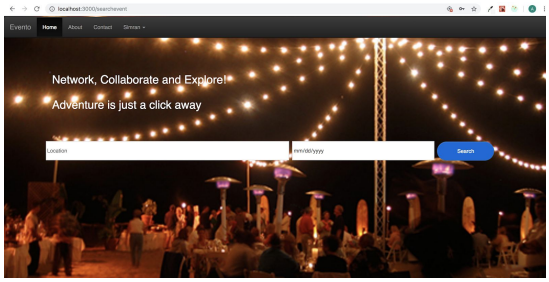


Fig. 8: Search Bar allows you to search for events

User can search for events based on location and date and view their details as follows.

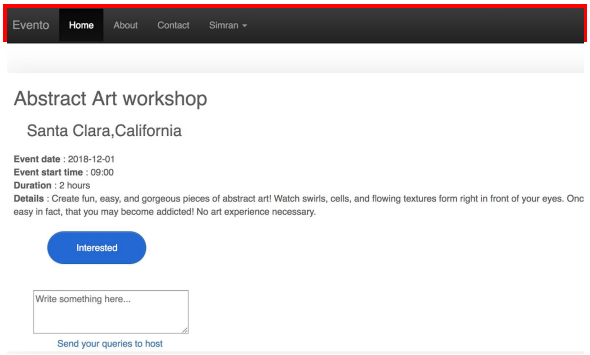


Fig. 9: Event details page

V. FUTURE ENHANCEMENTS

This application has wide scope of enhancements, a few of which are -

1. Currently, in the application, the user can only see the number of people who might be interested in an event.

We can add a suggestion module to prompt the user what changes he needs to make in order to get more people interested in his event.

2. Another feature that can be added is Google Calendar integration so that an event which the user is interested in gets marked in his calendar to make sure he never misses an event he is interested in.

3. Currently the prediction-recommendation is done on the basis of only two features-proximity and relevance. Richer features can be engineered for better prediction and recommendation.

VI. CONCLUSION

In conclusion, Evento will certainly help in improving the event management scenario. Evento can come handy in the domain of event planning and management. Through machine learning models, we built an application that suggests relevant events for users through the users data. Taking it a step further, we successfully predicted the number of potential interested users for any event which one might want to host. With more extensive data, the model can be trained rigorously to achieve results that can mimic the human preferences for attending events and in turn provide reliable prediction results.

VII. ACKNOWLEDGEMENT

Our sincere thanks to Prof. Rakesh Ranjan for his invaluable guidance. His comments and constructive feedback at every step was the key for successfully completing the project. The insights that he shared will definitely make the application useful and handy for event planners ,to plan their events as meticulously as possible, and also for people who want to quickly view recommendations.

VIII. REFERENCES

1.Event recommendation dataset from kaggle website.
https://www.kaggle.com/c/event-recommendation-engine-challenge

2.One of the experiences of a group on idea evolution on the kaggle dataset.
https://datathinking.wordpress.com/2013/02/10/event-recommendation-engine-challenge-kaggle/

3.Event recommendation challenge on kaggle discussion
http://webmining.olariu.org/event-recommendation-contest-on-kaggle/