# The Automated Travel Agent: Hotel Recommendations Using ML

Team Quark

Guide: Dr. Avinash Sharma
TA mentor: Tarun Gupta

Kushagra Chandak
Diplav Srivastava
Harshil Jain

# Problem Statement

-These days the ecommerce industry is rapidly increasing and consumers are increasingly interested in booking hotels online. So, if hotels are recommended to users, it'll save their time and increase the marketing of the e-commerce website.

-Recommending products to consumers is a popular application of ML, especially when there's substantial data about consumers' preferences.

-We've a dataset containing information about 37 million users and we need to recommend hotel clusters to these users.

# Dataset Information

Data Source - Kaggle

Number of Instances - 37 million

Number of features-23

# Dataset and Features

The data consisted of anonymized features including aspects such as number of children, number of rooms booked, destination visited, state of check in and check out, location of user when booking, whether or not the booking was part of a package, etc.

Some feature engineering was used on the dataset to extract more useful features. The date of check in, date checkout and date of booking were removed and replaced with length of stay -- dates discretized into a format which is more friendly to use with our learning algorithms.  Also, this feature is clearly more useful for our problem.

# Features

| Feature name | Description | Data type |
|---|---|---|
| site_name | ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...) | int |
| posa_continent | ID of continent associated with site_name | int |
| user_location_country | The ID of the country the customer is located | int |
| user_location_region | The ID of the region the customer is located | int |
| user_location_city | The ID of the city the customer is located | int |
| orig_destination_distance | Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated<br>int | double |
| is_mobile | 1 when a user connected from a mobile device, 0 otherwise | int |
| is_package | 1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise | int |
| channel | ID of a marketing channel | int |
| srch_adults_cnt | The number of adults specified in the hotel room | int |
| srch_children_cnt | The number of (extra occupancy) children specified in the hotel room | int |
| srch_rm_cnt | The number of hotel rooms specified in the search | int |
| srch_destination_id | ID of the destination where the hotel search was performed | int |
| srch_destination_type_id | Type of destination | int |
| hotel_continent | Hotel continent | int |
| hotel_country | Hotel country | int |
| hotel_market | Hotel market | int |

# Problem Challenges

***-Huge Dataset***

Since the training dataset was so huge( 37 million data point) and required a lot computational time one of the important task was to downsample data such that most of the useful information is retained.

***-Few attributes were non vectorial***

Few feature were of string form but had significant importance , so we converted that attribute into integer form.

***-Missing attribute values***

# Pre Processing

*-Replaced missing attributes with the most frequent value.*

*-Downsampling*

Downsampled the training dataset such that prior probabilities of all the hotel clusters remain same.

***-More useful feature  obtained***

Some of the feature like date in time and out time was converted into length_of_stay which is a more significant or a useful feature for our problem.

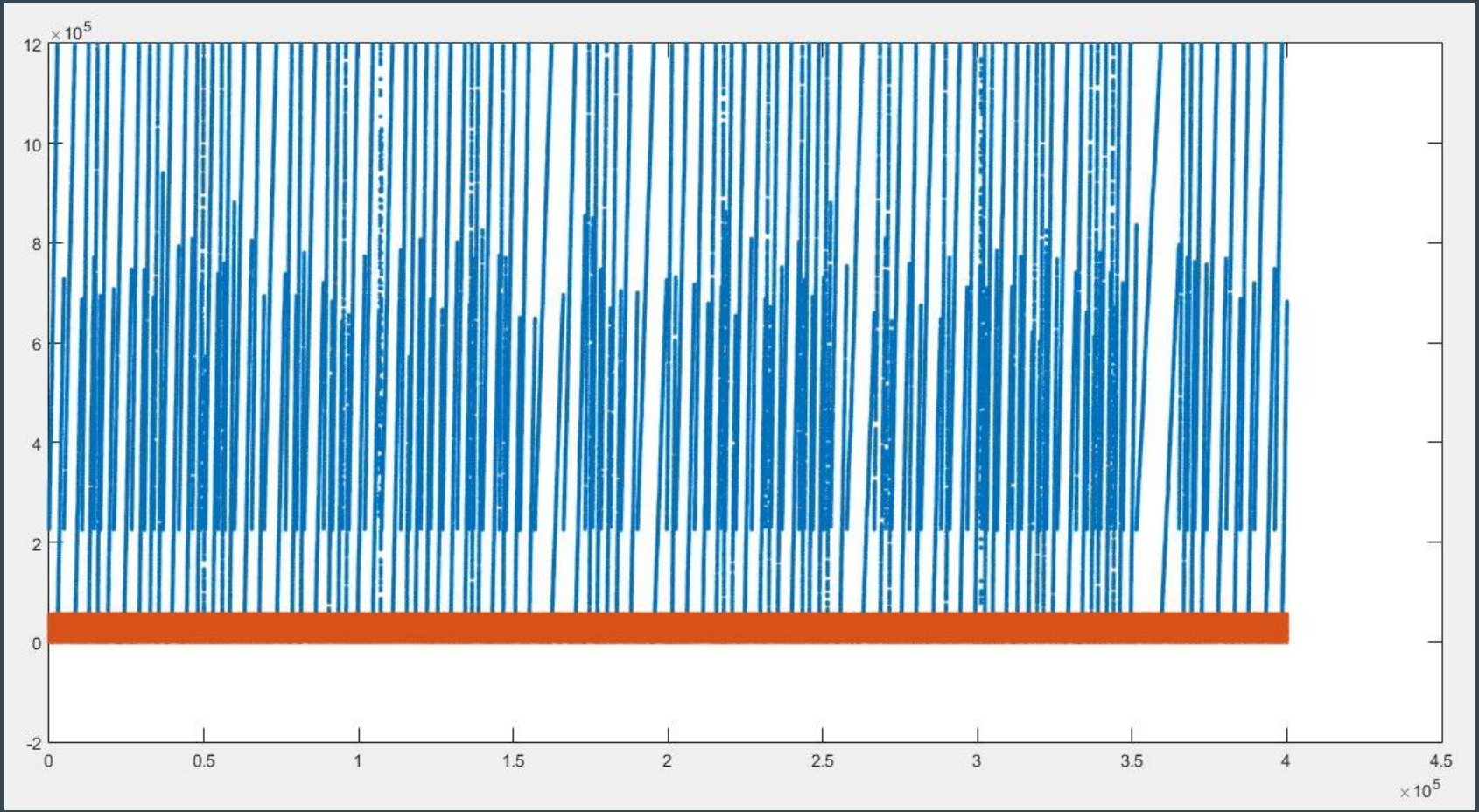
***-Performed PCA***

Performed PCA on the resultant training dataset to reduce the dimensions from 21 to 12.

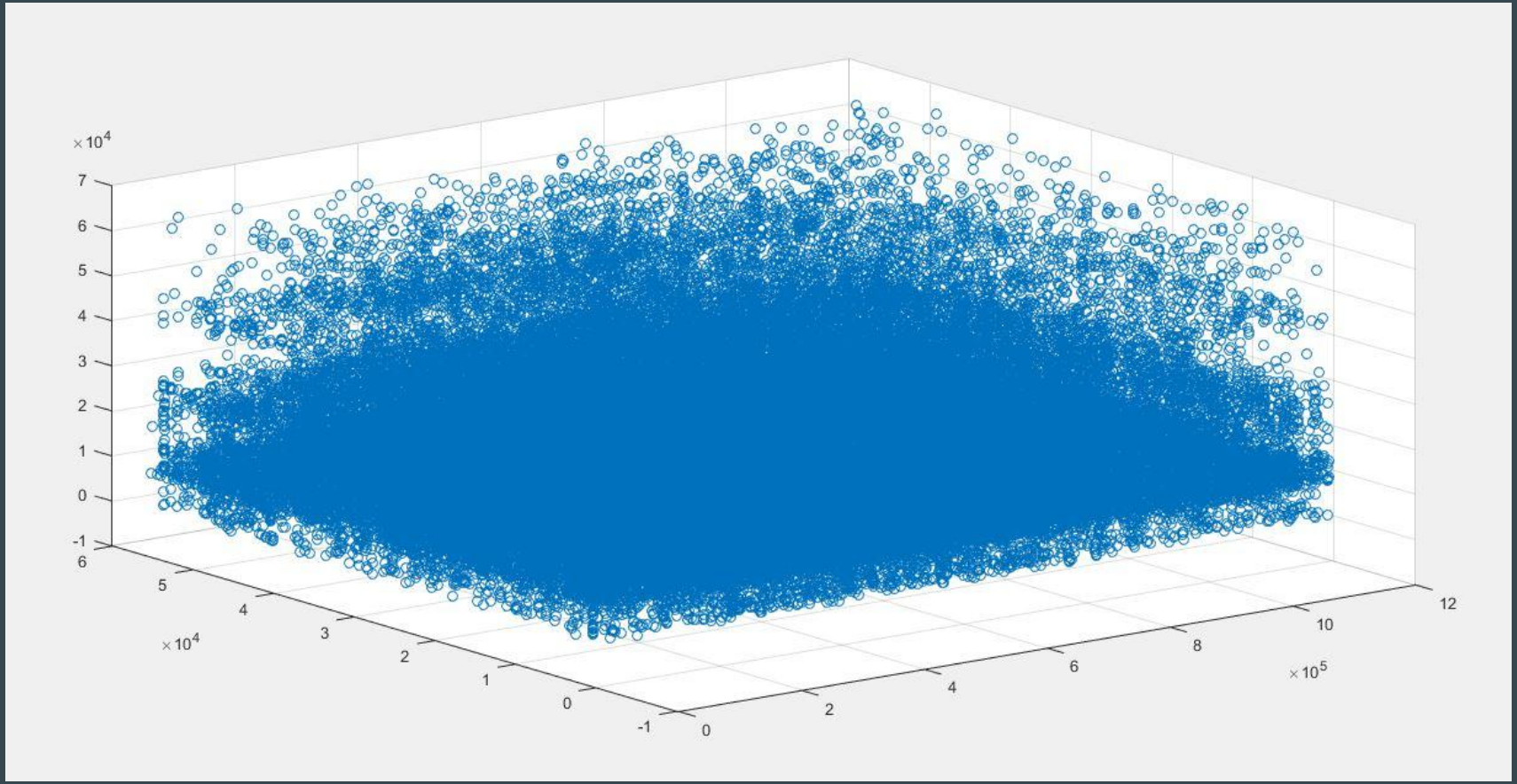
# Principal Component Analysis

# PCA

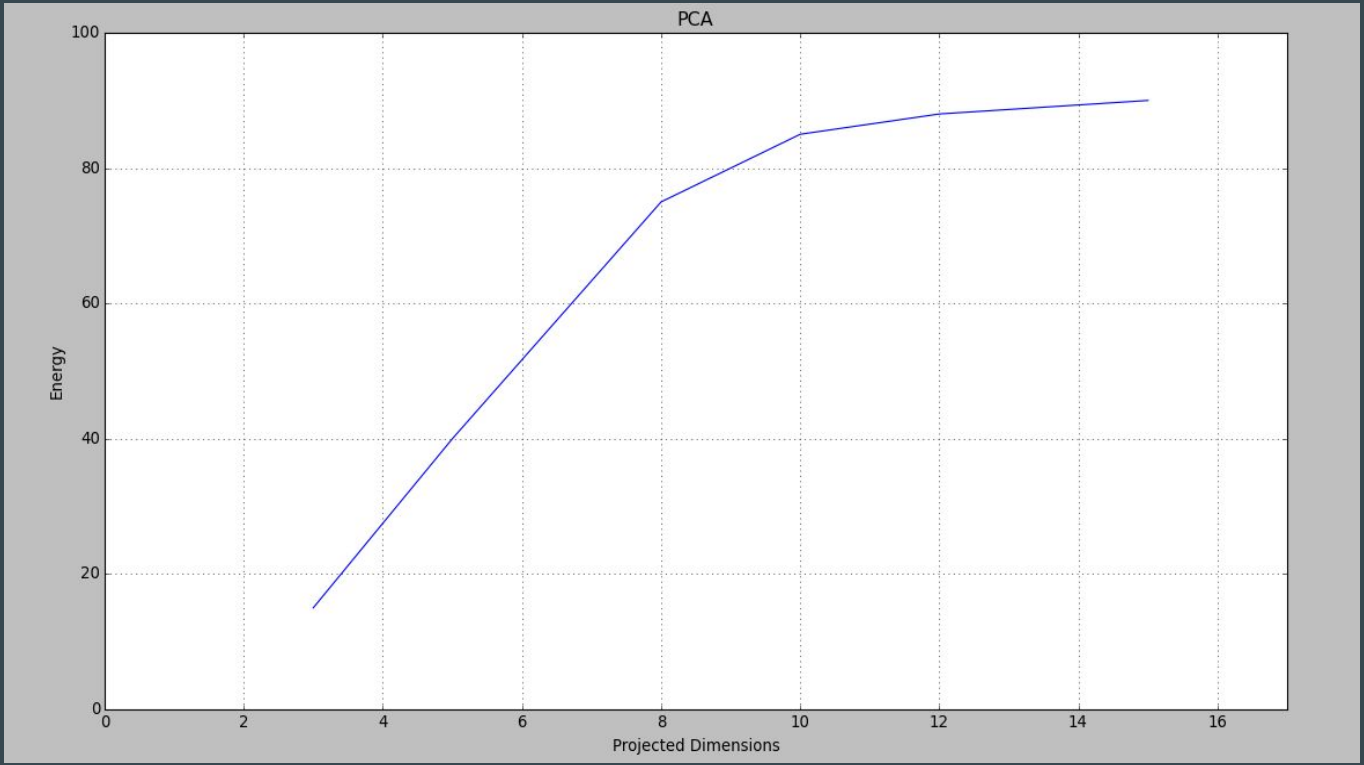
# Projected Data in 2-D



# Projected Data in 3-D



# Choosing the value of K (Projected Dimension)




$$\text{Energy} = \frac{\left(\sum_{i=1}^{K} \lambda_i\right)}{\left(\sum_{j=1}^{N} \lambda_j\right)}$$

# Algorithms Implemented

- Naive Bayes
- Softmax Regression
- SVM
- Weighted user similarity -- involves combination of clustering, kernelized similarity and weighting distance.

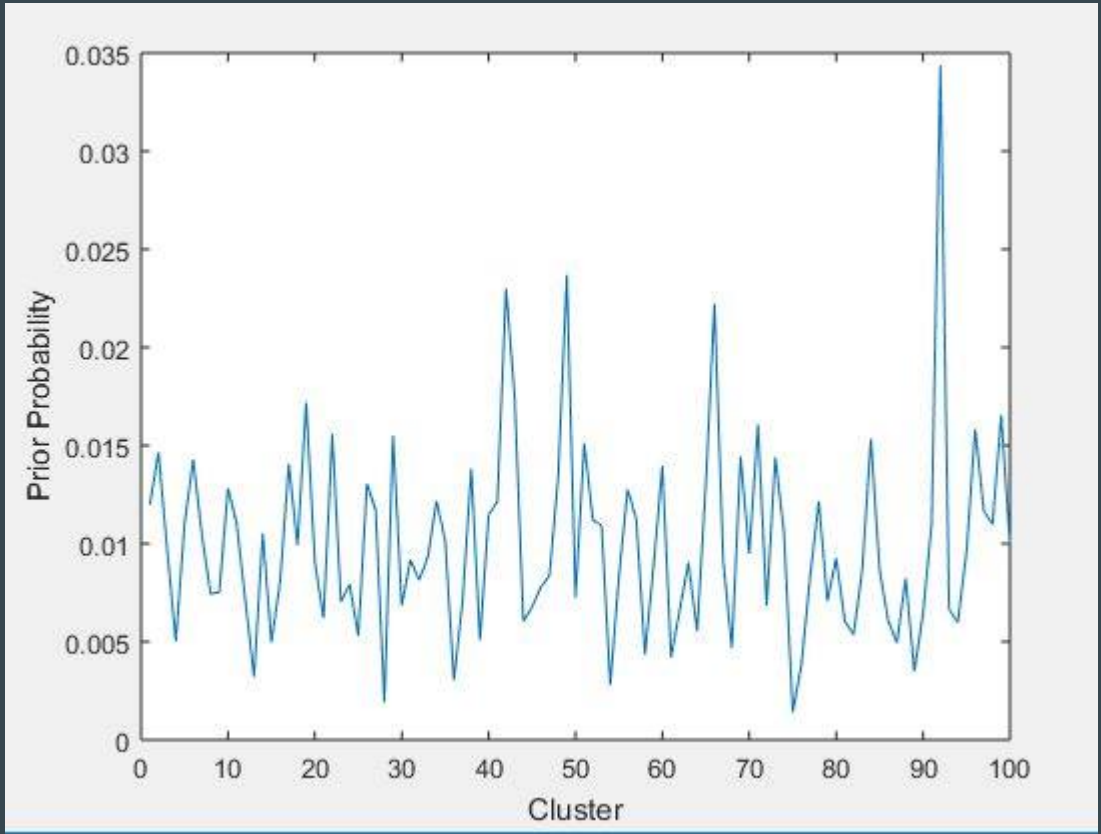
# Naive Bayes

...

# Analysis of Naive Bayes

For Naive Bayes, we hand-selected a few features, to get a basic machine learning algorithm running. We calculated target class probabilities for the categorical features, and  normalized continuous variables based on target label values. For each attribute in the test set, we converted its value to booking probability based on the the observed probabilities in the training set.

# Prior Probability of Cluster



# Assumptions

***Attribute Linearly Independent***

The main assumption made while applying Naive Bayes classifier was that attribute of training dataset obtained after PCA are linearly independent so that we can directly treat each attribute as independent random variable and multiply each of them to get probability of each cluster

***Normal Density Function***

Assumed normal density function for each class

# Results

| Algorithm | Precision | Recall | |
|---|---|---|---|
| Naive Bayes | 0.0596321 | 0.0567834 | |

# Support Vector Machines(SVM)

...

**Support vector machines** (**SVMs**) are supervised learning models with associated learning algorithms that analyze data used for classification It uses kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs.

# Support Vector Machine (SVM)

Applied Multi Class SVM using 3 different kernels

- Linear
- Polynomial
- RBF

One Vs. All Classifier is used

SVM is applied on data obtained after PCA to downsampled data (40 lakhs instances).

# SVM Linear

Linear kernel performed poorly due to lack of linear separability of data. Hence, they were discarded.

## Results

| Algorithm | Precision | Recall | |
|---|---|---|---|
| SVM linear | 0.034829 | 0.032112 | |

# (SVM) Polynomial

$$K(x, y) = (x^{\mathsf{T}} y + c)^d$$

***Effect of changing d (order of polynomial).***

Using a polynomial kernel was considered, but this was found to be prohibitively expensive in terms of computational time required.

# SVM Polynomial for d=4

## Results

| Algorithm | Precision | Recall | |
|---|---|---|---|
| SVM Polynomial | 0.06197 | 0.05945 | |

# SVM (RBF)

***Effect of changing gamma.***

Then applied SVM RBF Kernel over the downsampled data and observed the results by varying different values of gamma and C and ratio of training and testing data(K).

$$K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2), \gamma > 0$$

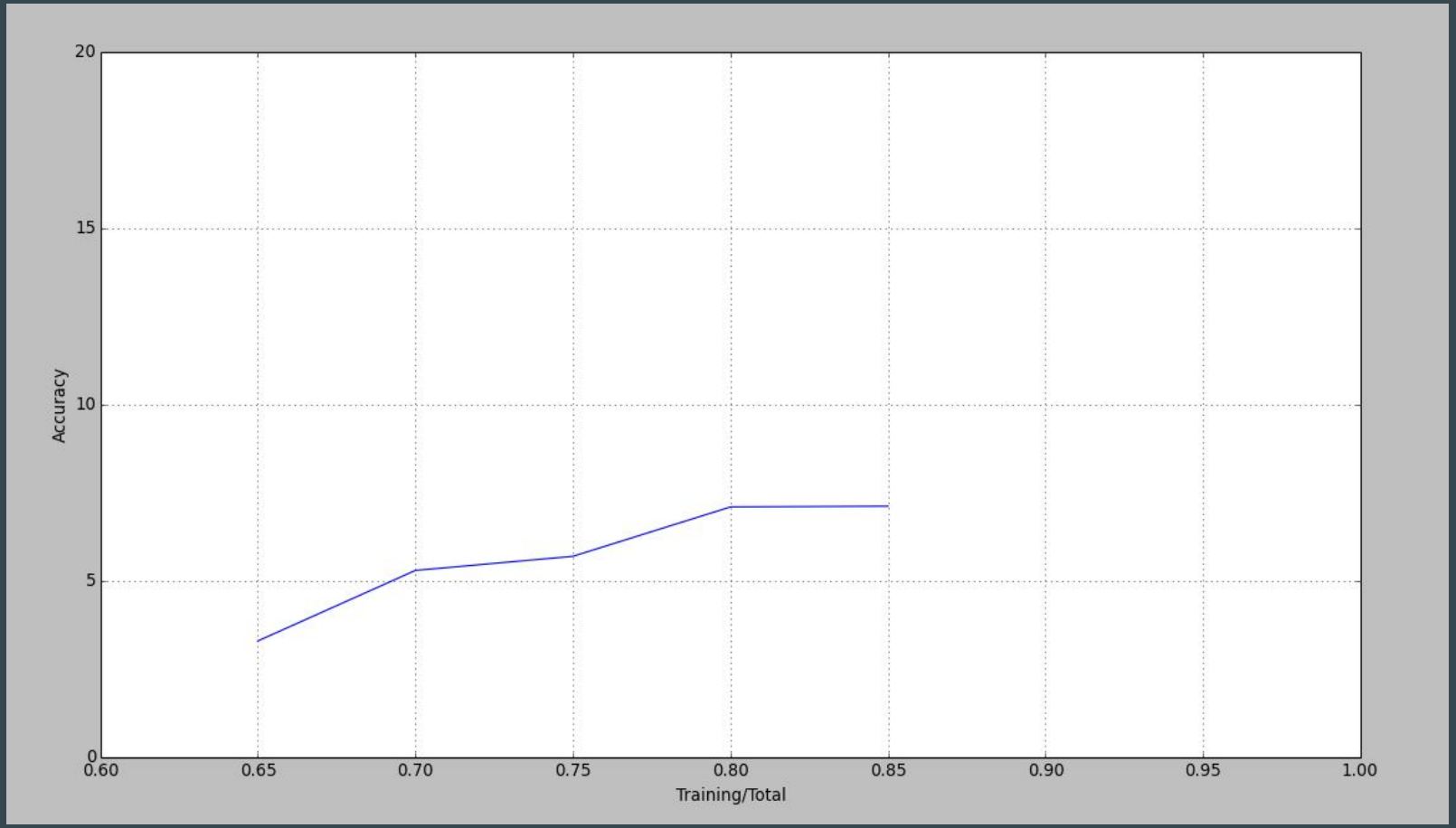
Gamma controls the shape of the "peaks" where you raise the points. A small gamma gives you a pointed bump in the higher dimensions (Low variance), a large gamma gives you a softer, broader bump(Higher variance)

# Results

## SVM RBF

| Algorithm | Precision | Recall | |
|---|---|---|---|
| SVM RBF | 0.073425 | 0.069846 | |

# K-Fold Validation(SVM)



# Softmax Regression

...

For each input vector we try to estimate its probability belonging to each of the class

# Softmax Regression

## Logistic Function

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

Parameter Vector

$$\theta = \begin{bmatrix} | & | & | & | \\ \theta^{(1)} & \theta^{(2)} & \cdots & \theta^{(K)} \\ | & | & | & | \end{bmatrix}.$$

# Softmax Regression

Probability that data point belong to label i

$$P(y^{(i)} = k|x^{(i)};\theta) = \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})}$$

Cost Function to be minimized

$$J(\theta) = -\left[\sum_{i=1}^{m}\sum_{k=1}^{K} 1\left\{y^{(i)} = k\right\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})}\right]$$

# Softmax Regression

***Parameter of gradient Descent***

Eta = 0.05

Threshold for each $|\Theta_k|$ = 0.1

Maximum Iteration=500

# Softmax Regression

## Results

| Algorithm | Precision | Recall | |
|---|---|---|---|
| Softmax | 0.05489 | 0.05154 | |

# Weighted User Similarity

-Training data clustered based on source destination id feature.

- Created a dictionary with src destination id as a key and the users in the training data that contain that id as a feature as the value.

-For each member of the test set, we hash into the group that share the same src destination id feature using the created dictionary. Once we have the group of training users, we then create a kernel matrix (vector) using some kernel function and find, say, top 50 users in the group.

-Once we have these top 50 users, we give each user a score based on the similarity, and for each hotel cluster sum up the scores of the users in the top 50 who booked that cluster. The cluster with the  highest score is recommended for the user.

# Measuring User Similarity

-Cosine similarity performed poorly, as certain features serve as numeric representations of qualitative information. (Eg. continent_id)

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Opted for Jaccard similarity which essentially counts the number of features shared.

$$jac(x, y) = \frac{|x \cap y|}{|X| + |y| - |x \cap y|}$$

# Measuring User Similarity

-We define the kernel matrix to be the vector K where the ith element of K is the result of the Jaccard kernel function. We then sort K and extract the top 50 users.

-Now that we have the sorted top 50 users and their similarity scores for each according to the kernel function, we need to assign each *hotel cluster* a score based on the number of times it appears in these top 50 users and how similar these users are.

-Lets define V to be a vector with 50 elements containing the hotel clusters (in order) of the top 50 users.  Score function:

$Score_1(cluster) = \sum_i \{V_i=cluster\} \exp(-i^2/2\tau^2)$ , $\tau$ = constant = 50.

# Measuring User Similarity

The basis of the equation above is to weigh users who are really close to the test user more than the users who are far away, but still heavily weigh clusters that appear more often in those top users.

# Results

| | Precision | Recall | |
|---|---|---|---|
| Cosine Similarity | 0.1557 | 0.1499 | |
| Jaccard Similarity | 0.1685 | 0.1656 | |

# Conclusion and Future Scope

-This project provides a good example for applying ML algorithms on a large dataset which lack obvious structure.

-Jaccard similarity here is better similarity criterion than cosine.

-Effect of various parameters on SVM kernels.

-Simple models like Naive Bayes and Logistic Regression can also give reasonable results.

-Gradient Boosting algorithms can also be used.