Machine Learning based Prediction of Confirmation of Waitlisted Indian Rail Tickets

Jyoti Aneja
janeja2@illinois.edu
Deptartment of Physics
UIUC

Anand Bhattad
bhattad2@illinios.edu
Department of CEE
UIUC

Shubhang Goswami
sgoswam3@illinois.edu
Department of Physics
UIUC

December 14, 2016

Abstract

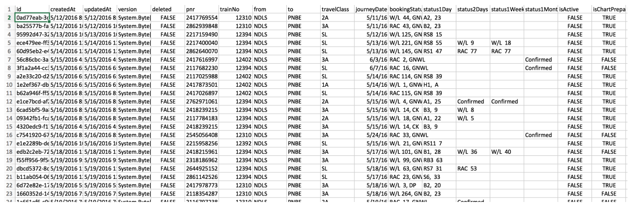
The Indian Rail supports 13 million passengers every day, and yet millions do not avail confirmed tickets for their travel. In this project, we have used machine learning on data obtained from a mobile application to predict the confirmation of wait-listed train tickets. We have used Scikit-Learn to train soft SVM, Random Forest and Naive Bayes models and compared them with the results of a neural net trained using Tensor Flow. From our experiments, we conclude that it is possible to learn complicated features that govern the confirmation of rail tickets. The learned model can then be used to predict if (and when) a wait-listed ticket will get confirmed.

1 Introduction

Tens of millions of people use the Indian Railway system each day. Yet, a large fraction of them are unable to obtain a confirmed ticket before their travels. While some travel (illegally) on wait-listed tickets, others cancel their tickets at the very last moment. The allocated tickets for most trains get sold-out during a very short window after booking opens. The situation gets even more acute during the festive season and holidays. Reserving a ticket on a wait-list is particularly inconvenient since currently there are no platforms that offer any insights into the chances of a wait-listed ticket getting confirmed. In this project, we study the feasibility of making such predictions using the data from past railway travels. Empowered with this knowledge, users could be in a better position to make further decisions—either wait for the ticket to get confirmed, or make alternative travel arrangements. We obtain our data from a mobile application, carefully select the features that could affect confirmation of a wait-listed ticket, and design rules (e.g. interpolation) to improve the quality of our final data-set. We then train linear and non-linear machine learning models and do inference on held-out test data. The scope of this problem and the scale at which a good solution can benefit millions was our motivation for us to take up this as a project.

1.1 Dataset

We got our data from a mobile application[3, 2] that stores passengers' information when they query to check the current status of their wait-listed ticket, or to store their travel details. We have almost 10,000 travel locations and over 900,000 entries recorded, as shown in Fig.1 There are eight travel classes available—1A, 2A, 2S, 3A, 3E, CC, FC, SL. The data includes the status of the wait-listed ticket at different time intervals before the date of the journey. Specifically, we have information on the status of the ticket 1 day, 2 days, 1 week and 1 month before the travel. This gives insights into the rate of cancellation on the



	id	createdAt	updatedAt	version	deleted	pnr	trainNo	from	to	travelClass	journeyDate	bookingStatu	status1Day	status2Days	status1Week	status1Mont	isActive	isChartPrepa
2	0ad77eab-3c	5/12/2016 8:	5/12/2016 8:	System.Byte[	FALSE	2417769554	12310	NDLS	PNBE	2A	5/11/16	W/L 44, GN\	A2, 23				FALSE	TRUE
3	ba25577b-fa	5/12/2016 1(	5/12/2016 1(	System.Byte[	FALSE	2862939848	12310	NDLS	PNBE	3A	5/11/16	RAC 43, GN\	B2, 23				FALSE	TRUE
4	95992d47-32	5/13/2016 1(	5/13/2016 1(	System.Byte[	FALSE	2217159490	12394	NDLS	PNBE	SL	5/12/16	W/L 125, GN	RS8 15				FALSE	TRUE
5	ece479ee-ff3	5/14/2016 1:	5/14/2016 1:	System.Byte[	FALSE	2217400040	12394	NDLS	PNBE	SL	5/13/16	W/L 221, GN	RS8 55	W/L 9	W/L 18		FALSE	TRUE
6	60d95eb2-e4	5/14/2016 1:	5/14/2016 1:	System.Byte[	FALSE	2862640070	12394	NDLS	PNBE	SL	5/13/16	W/L 145, GN	RS1 47	RAC 77	RAC 77		FALSE	TRUE
7	56c86cbc-3a	5/15/2016 4:	5/15/2016 4:	System.Byte[	FALSE	2417616997	12402	NDLS	PNBE	3A	6/3/16	RAC 2, GNWL				Confirmed	FALSE	FALSE
8	3f1a2a44-cc3	5/15/2016 6:	5/15/2016 6:	System.Byte[	FALSE	2117682230	12394	NDLS	PNBE	SL	6/7/16	RAC 16, GNWL				Confirmed	FALSE	FALSE
9	a2e33c20-d2	5/15/2016 6:	5/15/2016 6:	System.Byte[	FALSE	2117025988	12402	NDLS	PNBE	SL	5/14/16	RAC 114, GN	RS8 39				FALSE	TRUE
10	1e2ef367-db	5/15/2016 6:	5/15/2016 6:	System.Byte[	FALSE	2417873501	12402	NDLS	PNBE	1A	5/14/16	W/L 1, GNW	H1, A				FALSE	TRUE
11	b62a946f-ff5	5/15/2016 8:	5/15/2016 8:	System.Byte[	FALSE	2417026897	12402	NDLS	PNBE	SL	5/14/16	RAC 115, GN	RS8 39				FALSE	TRUE
12	e1ce7bcd-af	5/16/2016 8:	5/16/2016 8:	System.Byte[	FALSE	2762971061	12394	NDLS	PNBE	2A	5/15/16	W/L 4, GNW	A1, 25	Confirmed	Confirmed		FALSE	TRUE
13	6cad5bf5-9a	5/16/2016 8:	5/16/2016 8:	System.Byte[	FALSE	2418239215	12394	NDLS	PNBE	3A	5/15/16	W/L 14, CK	B3, 9	W/L 8			FALSE	TRUE
14	09342fb1-fca	5/16/2016 8:	5/16/2016 8:	System.Byte[	FALSE	2117784183	12394	NDLS	PNBE	2A	5/15/16	W/L 18, GN\	A1, 22	W/L 5			FALSE	TRUE
15	4320edc9-f1	5/16/2016 4:	5/16/2016 4:	System.Byte[	FALSE	2418239215	12394	NDLS	PNBE	3A	5/15/16	W/L 14, CK	B3, 9				FALSE	TRUE
16	c7541920-67	5/16/2016 8:	5/16/2016 8:	System.Byte[	FALSE	2545056408	12310	NDLS	PNBE	3A	5/24/16	RAC 33, GNWL				Confirmed	FALSE	FALSE
17	e1e2289b-de	5/16/2016 1(	5/16/2016 1(	System.Byte[	FALSE	2215958256	12392	NDLS	PNBE	SL	5/15/16	W/L 21, GN\	RS11 7				FALSE	TRUE
18	edb2c2eb-72	5/18/2016 1:	5/18/2016 1:	System.Byte[	FALSE	2418215961	12394	NDLS	PNBE	3A	5/17/16	W/L 101, GN	B1, 28	W/L 36	W/L 40		FALSE	FALSE
19	f55ff956-9f5	5/19/2016 9:	5/19/2016 9:	System.Byte[	FALSE	2318186962	12394	NDLS	PNBE	3A	5/17/16	W/L 99, GN\	RB3 63				FALSE	TRUE
20	dbcd5372-8c	5/19/2016 1:	5/19/2016 1:	System.Byte[	FALSE	2644925152	12394	NDLS	PNBE	SL	5/18/16	W/L 63, GN\	RS7 31	RAC 53			FALSE	TRUE
21	b11ab054-06	5/19/2016 1:	5/19/2016 1:	System.Byte[	FALSE	2861142526	12394	NDLS	PNBE	SL	5/17/16	RAC 23, GN\	S6, 33				FALSE	TRUE
22	6d72e82e-17	5/19/2016 5:	5/19/2016 5:	System.Byte[	FALSE	2417978773	12310	NDLS	PNBE	3A	5/18/16	W/L 3, DP	B2, 20				FALSE	TRUE
23	1660352d-14	5/19/2016 7:	5/19/2016 7:	System.Byte[	FALSE	2118354287	12310	NDLS	PNBE	3A	5/18/16	W/L 264, GN	B2, 23				FALSE	FALSE

Figure 1: Snippet of the Data

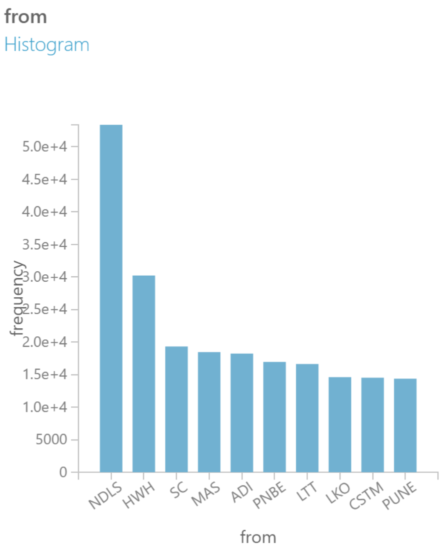


Figure 2: Stations "from" where highest number of trains leave

confirmed tickets. The data has the following limitations, which we either circumvent or overcome in our project:

1. *Data Sparsity:* There are numerous instances where features are missing because of the design of the mobile application. Due to this, we focused on two popular classes of travel SL and 3A. We selected as features the travel class, the day of travel, and the ticket status 2 days, 1 week and 1 month before the travel. In section 2 we describe in detail our feature engineering process.

2. *Missing true labels:* The booking status (which is the status on the day the ticket was booked) is available for all the data, but the confirmation status is not. Therefore, for simplicity we use the status 1 day before the travel to determine the confirmation status.

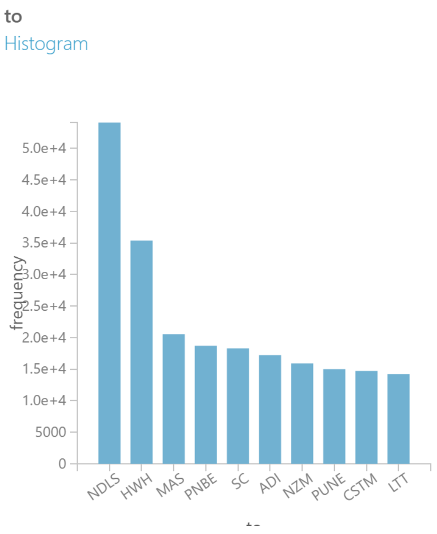


Figure 3: Stations "to" where highest number of trains go

.

2 Feature Engineering

Firstly, we figure out the stations with heavy traffic (Fig.2 and Fig.3). Next, by plotting the status of the wait-listed ticket with time (Fig.4), we observe that region from 1-month to 2-days can approximated with a linear drop. Also, the region from 2-days status to 1-day status mimics a truncated logarithmic.

We devised a set of rules to interpolate values in cases where a few features were missing. We focus on data related to the booking status and the status of the wait-listed ticket 1 day, 2 days, 1 week and 1 month before the journey. Our interpolating is restricted to these 5 values. The rules are as follows:

1. If **3 out of the 4** points are missing then we exclude that data point.

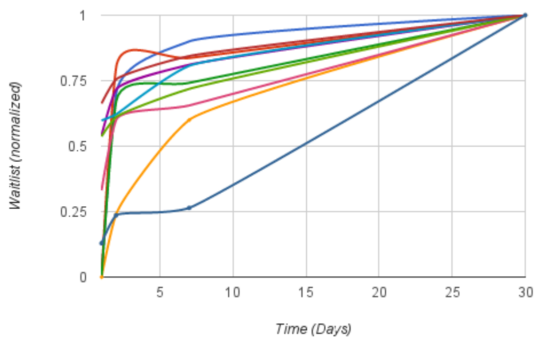


Figure 4: Data vs Time for Interpolation

2. If **2 out of the 4** points are missing
   (a) *1 month and 1 week status are missing:* If (booking status - status 2-days) ≥ 60, then set 1 month status to booking status, else set 1 week status to booking status, and interpolate the remaining point by fitting a straight line.
   (b) For all the other cases, the given data points were used to fit a straight line, and the missing points were interpolated.
3. If **1 out of the 4** points is missing
   (a) *1 month status is missing:* Use max(1-week, Booking Status) as the 1-month status.
   (b) For all the other cases, the available data points were used to fit a straight line, and the missing points were interpolated.
4. For fitting 1-day status, a truncated logarithmic interpolation was used.

3 Experiments

3.1 Support Vector Machines

Hard SVM[1] performed poorly on testing data but performed quite well on the training. We saw that the accuracy of prediction increased with the number of iterations (Fig.5), and decreased with the slack variable value C (Fig.6). We also ran soft SVM with the constant $C = 1.0$ and learning rate 0.0001 for 100 iterations. We performed experiments on different routes, starting with the route with the heaviest traffic (Fig.2). We repeated this on five different routes. The results for soft SVM are shown in Fig.10.

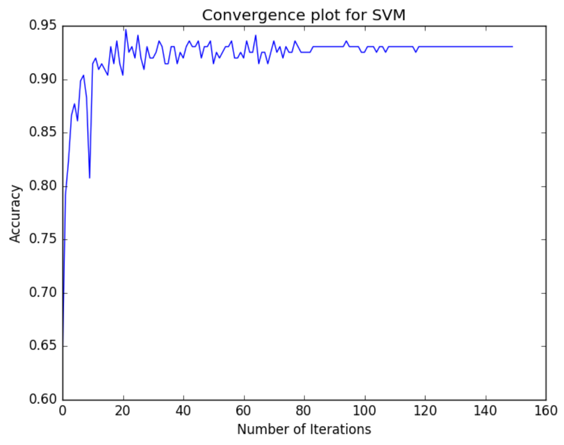


Figure 5: Convergence Plot for SVM over Delhi-Patna Route

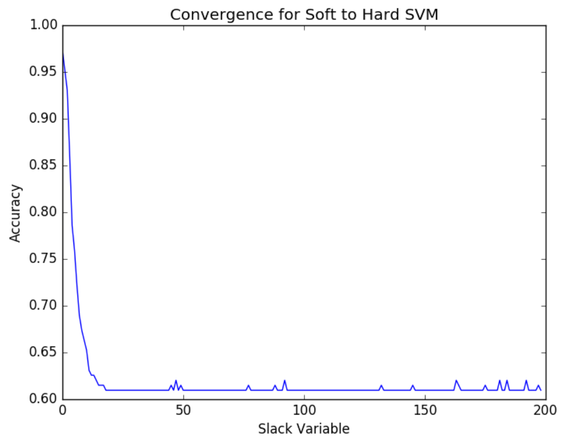


Figure 6: Soft to Hard SVM

3.2 Neural Networks

We trained a neural network with two hidden layers (Fig.7.[1]) using Tensor Flow. The 12 nodes of the input layer correspond to the 12 input features. For the first hidden layer we used 25 nodes, and for the second layer we used 5 hidden nodes. After parameter tuning, the learning rate of 0.0005 and 0.0001 produced good results for our model. We used 0.0001 as our learning rate, and square-loss as the output error function. We evaluated with `tanh`, `ReLU` an `sigmoid` non-linearity after every layer and found `sigmoid` to perform the best. Using `tanh`, the accuracy was about 40%, `ReLU` gave us accuracy of about 65% and `sigmoid` gave accuracy of 80%

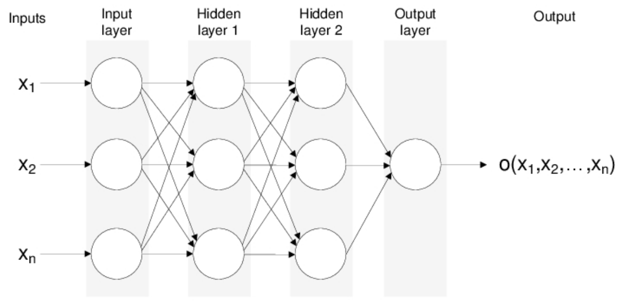


Figure 7: Neural Network Architecture

3.3 Other Experiments Performed

1. For logistic regression we achieved an accuracy of only 45%. This was understandable as some of our features (like status 2 days, 1 weeek and 1 month before travel) are integers, and others like travel class and the day of travel are binary (one-hot vectors).

2. *Station Clustering:* We grouped all nearby stations into one big logical station. For example New Delhi, Hazrat Nizamuddin, Anand Vihar and Faridabad were bundled into one. With this approach, the performance of SVM improved by 2% to 3%, although there was no perceptible difference with neural nets.

[1]`http://www.intechopen.com/source/html/38302/media/ann.jpg)`

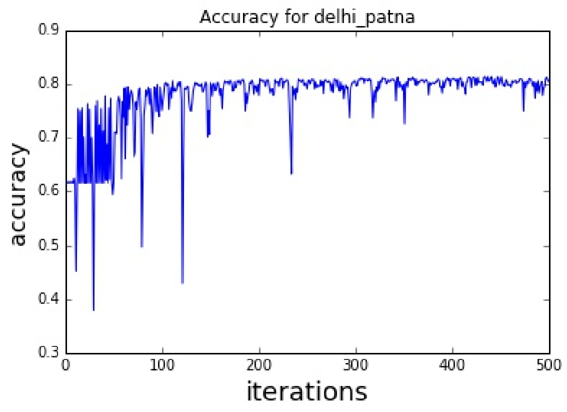


Figure 8: Convergence plot for Neural Network on Delhi Patna Route

3. Reducing features by replacing the travel-day one-hot vector with just information of weekday or weekend. With this, the performance dropped marginally. For SVM, the accuracy for New Delhi-Patna route was 84% and using Neural Network it was 71%.

4. Reducing one hidden layer disturbed the performance of our model and increasing one additional hidden layer did not improve performance significantly with our data set

5. We directly used Naïve Bayes model from *Scikit-learn*. As per our expectations, Naïve Bayes performed extremely poorly. Our time series features are dependent on each other, therefore, the conditional independence assumption is not valid for our data set and hence poor accuracy.

6. Random Forest from *Scikit-learn* library was also tried and for 10 trees, the results were not as good as SVM.

7. We tried using principal component analysis by reducing features space from 12 to 2 and also from 12 to 3 to see if they are linearly separable in these dimensional planes. The following figures clearly indicate that reducing dimensionality did not result into linear separability for both these cases.

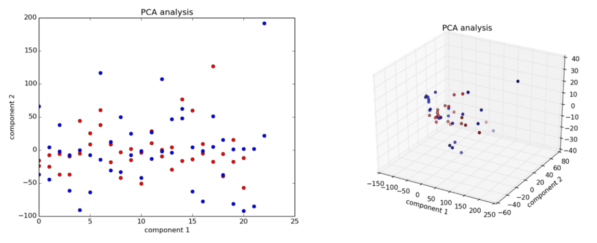
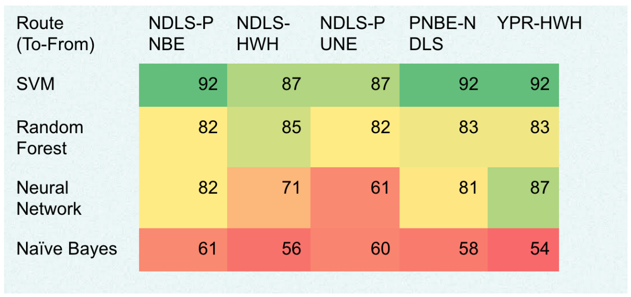


Figure 9: 2D-3D PCA Plot

3.4 Summary of Results

Fig. 2 and Fig. 3 helped us to validate our models on different routes for which we had enough data to train effectively. We tested our models for the routes: Patna - New Delhi, Howrah (Kolkata) - Yeshvantpur (Bangalore), New Delhi-Ahmedabad, New Delhi-Pune, Pune -New Delhi, New Delhi-CSTM (Mumbai) and Chennai-Secunderabad (Hyderabdad). Results of a selected few are listed in the accuracy map in Fig. 10.



Route (To-From)	NDLS-PNBE	NDLS-HWH	NDLS-PUNE	PNBE-NDLS	YPR-HWH
SVM	92	87	87	92	92
Random Forest	82	85	82	83	83
Neural Network	82	71	61	81	87
Naïve Bayes	61	56	60	58	54

Figure 10: Accuracies of Various Methods Over Major Routes [NDLS = New Delhi, PNBE = Patna, PUN = Pune, YPR = Yeshvantpur (Bangalore), HWH =Howrah (Kolkata)]

4 Conclusion and Discussion

1. From the experiments we performed, soft SVM was the clear winner.
2. We realized that the 'day of travel' is an important feature. The accuracy dropped to 60% when we ignored it.
3. From the local clustering of the routes, we observed that the close-by cities when clustered together as a single group yielded similar or better accuracy. Therefore, to scale the model for all possible routes, the ideal way would be to consider locations using such local clustering.
4. Feature Engineering is the most important component of Machine Learning and getting the right features is the key to develop robust and predictive algorithms. As is it rightly said in the ML community, humans do feature engineering, computers do feature learning.

5 Future Work

1. Our interpolation trick to fill the missing data leaves us with a relatively small number of data points that can be used to train a good model. This leads to poor performance especially in neural networks. It will be interesting to find a way to fill missing points without disturbing the inherent patterns and at the same time not lose a lot of data.
2. Currently, we are only recording 1day, 2day, 7day and 30day wait-list data, if we can get more such timed data then it will be possible to approximate them with a distribution and use Expectation Maximization algorithm to find most likely parameters to fill in the missing points.
3. This project can be extended to suggest an alternate route to a user who has a low chance of confirmation on a wait-listed ticket. This part needs some study of combinatorial optimization and we intend to pursue this in future.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[2] *IndianRail Windows Apps on Microsoft Store*. `https://www.microsoft.com/en-us/store/p/indianrail/9wzdncrfj26c`.

[3] PB.Appetite. *Indian Rail Train Status - Android Apps on Google Play*. `https://play.google.com/store/apps/details?id=com.pb.indianrail&hl=en`.