



FLY HIGH

ENCP7250 Engineering Big Data systems

Abstract

A flight fare prediction to help identify the best date to travel on the cheapest fare.

Arvind Venkatasubramanian, Zalak Shah, Abhijeet Singh,
Shanmughasudan Veerabahu, Ashish Maheedhara

Contents

OBJECTIVE	2
Introduction	2
Scope of Project	2
Key Facts	2
WORK ALLOCATION	3
DATASET	3
Dataset Details	3
DATA CLEANING AND PREPROCESS	5
NORMALIZATION OF DATA	5
MACHINE LEARNING ALGORITHMS	6
Feature Selection	7
Multiple Linear Regression	9
Why Linear Regression?	9
Running of Linear Regression	10
LINEAR REGRESSION RESULTS	10
Challenges Faced in Linear Regression	12
NEURAL NETWORK	12
Why Neural Network?	12
Neural Network in Azure	12
Model Setup Steps	13
NEURAL NETWORK RESULT	15
Neural Network Challenges	16
PREDICTIONS	16
HBASE DATABASE SETUP	17
Challenges Faced in HBASE	17
HBASE connection using JAVA	20
USER INTERFACE	20
CHALLENGES FACED IN UI	21
FUTURE SCOPE	23
BIBILOGRAPHY	24

OBJECTIVE

Introduction:

Everyone requires to travel one time or the other. But sometimes the travel plans do not work out because of various reasons. One of the most important reason is Air Fares and travel tickets. There is no clear pattern which would help the customer to pick the best date to travel with the cheapest price. Similarly, there is no easy way to identify what is the best date for a customer to book the ticket.

Airlines promote dynamic pricing of their tickets, or a kind of pricing based on demand and supply. The airlines have full control on the ticket prices, and based on their profit margin or sales decide on the price of the tickets.

At this point, the customer is dependent on the airline to purchase a cheap ticket. Many a times, the airlines do a steep increase in the prices based on a number of features like number of unsold seats or frequency of the flight between the source and the destination. There is not really a clear pattern in the price of the tickets. However, given a data for a year of airlines and ticket prices, we can predict the prices for the next year.

In order to make it easier for the customer, we propose an airline ticket prediction system. There is a prediction on the ticket fare. This ticket fare would help identify the cheapest date to travel or even the cheapest date to book a ticket. This way the customer would know that they would be able to save the maximum amount of money, if they have flexible dates to travel or are willing to take a chance by booking at a future date.

Scope of Project

- a. Building a prediction for best booking dates.
- b. Building a prediction for best travel dates.

Key Facts

- There is base fare for each flight. This base fare has all the considerations of the fare for the airlines.
- Ticket Fare is equal to base fare plus a fluctuation amount. This fluctuation is defined as the delta of price change. There are various factors which can affect the fluctuations
 - Oil Prices
 - Number of Unsold seats
 - Distance between the source and the destination
 - Frequency of flights between origin and destination
 - Dates of travel
- The consideration is for US domestic flights only as available for free download provided by the 'Bureau of Transportation Statistics' (<http://www.transtats.bts.gov/>).

- Multiple flights ply between multiple origin and destination combination on the same dates.

WORK ALLOCATION

Name	Work Allocation
Arvind	Data Preprocess and Cleaning
Zalak	Machine learning Algorithms
Abhijet	Hbase setup and retrieval of data
Ashish	User Interface interaction with Hbase
Sudan	Hbase setup on EC2 and local standalone

DATASET

The dataset obtained was from the official United States Bureau of Transportation, where we selected the aviation data. There were 3 datasets that were taken into consideration. We had split the data into quarterly data, to allow easy slicing and dicing of the preprocess data.

However, the training data under consideration is the **Complete 2015 data** with the fares, origin and destination details.

Dataset Details

- a. Market Data (http://www.transtats.bts.gov/Fields.asp?Table_ID=247)

The dataset consists of the Market data of all the airlines for a given year. For our training dataset we have considered the year to be 2015. The columns of the dataset are

- YEAR
The year for which the data is collected.
- Quarter
The quarter for the data. Since it is a yearly data, its value is ranging from 1-4.
- Ticket Carrier
The consideration is for the carrier for which the tickets are sold. We faced a challenge in deciding between Ticket Carrier and Operator Carrier. However, a number of airlines have a 'codeshare' agreement, where for a ticket, we have another carrier flight which operates for them. Hence we went ahead with Ticket Carrier.

- Market Fare
This is the ticket price. This was renamed to 'Ticket Fare' for easier understanding and segregation between Base fare and ticket fare.

b. All carrier data (http://www.transtats.bts.gov/Fields.asp?Table_ID=311)

This dataset consists the details of all the carriers between an origin and destination combination and their date of travel. It also contains the details about the carrier for that particular day. The most important feature from this dataset is the base fare.

- SEATS
The number of seats that are unsold.
- ORIGIN
The origin city details and abbreviation
- ORIGIN_CITY_NAME
Name of the city
- ORIGIN_STATE_ABR
- ORIGIN_STATE_NM
- DEST
Destination details
- DEST_CITY_NAME
- DEST_STATE_ABR
- DEST_STATE_NM
- ORIGIN_AIRPORT_ID
- DEST_AIRPORT_ID
- DATE_OF_BOOKING
- DATE_OF_TRAVEL
- BaseFare

c. Master Coordinates (http://www.transtats.bts.gov/Fields.asp?Table_ID=288)

This dataset is to compare and capture the latitude and longitude details based on the Airport code. The reason for capturing this data is to calculate the distance.

- AIRPORT_ID
- LATITUDE
- LONGITUDE

This allowed us to get the distance during the datacleaning process.

DATA CLEANING AND PREPROCESS

Data cleaning is one of the most important aspects before starting any data projects. Since the dataset is very huge, it was important to split the data into multiple sets and perform the data cleaning steps and actions.

We had specifically used R Script to perform the data cleaning process. This is because R consists of a lot of flexibility in slicing and dicing of the data and allows easy writing to the csv to the end record. The following were the data cleaning steps that were followed

- Removing of the NA and NULL values
- Combining the multiple csv files based on a common key
Since there are different files and we require to join them based on a common key (Carrier), we are able to combine them together.
- Split to the latitude and longitude of origin and destination data by comparing it with the Coordinate data.
- Split into Training and Test Data

Train Data – Data for year 2015

Test Data – Data for Q1 and Q2 of 2016

- Split the date of travel to corresponding Day of the week and Month of the Year.
- Extracted the distance by checking between the coordinates of origin and destination.

NORMALIZATION OF DATA

Normalizing is one of the most important steps for any data project. We require to normalize the data in order to apply any machine learning algorithm further ahead. It is also applied in order for any model to try and understand relations between the various factors of the dataset.

In our dataset, we have performed normal standardization using R Script package clusterSim.

This package allows the luxury of choosing the type of algorithms at the syntax level itself.

(<http://stackoverflow.com/questions/15215457/standardize-data-columns-in-r>).

```
| data_read_2$Base_Fare_Normalized <- data.Normalization(data_read_2$BaseFare,type ="n1",norm
data_read_2$Month_Standardization <- data.Normalization(data_read_2$MONTH_NUMBER,type ="n3"
data_read_2$DAYS_Standardized <- data.Normalization(data_read_2$DAYS,type ="n3",normalizati
data_read_2$Seats_Normalized <- data.Normalization(data_read_2$SEATS,type ="n3",normalizati
```

This standard data was converted with the idea for our prediction algorithm. The initial plan was to run Neural network. Neural network requires the input to be normalized within a range. It will not take text inputs. For this, we had to convert the following columns which we had tried to feed as an input to our prediction model

- Seats

Seats was normalized using the standard method. $((x - \text{mean}) / \text{sd})$

- Origin/Destination

For the origin, we had created a file which contains the list of all the source and destination (location list). All the airport locations were assigned with a random value in a sequence. This sequence was then standardized using the standard method of normalization.

- Base Fare

Base fare is one of the most important features that is required for our model prediction. The base fare that was provided was also standardized in the standard method.

- Seats

The number of available seats is also considered to be an important feature for the prediction.

- Carrier Frequency

There was a case in which one carrier would occur multiple number of times between origin and destination combinations. Hence, we scaled the input and assigned a frequency. Frequency would be the number of times it is occurring in the dataset.

Once the frequency is calculated, it was converted to a normalized form.

Additionally we had also tried to convert it to a Boolean value. However since the value was >10000 for a number of carriers, Excel or R does not let it allow for such high values. Hence we stuck to Normalized form

```
carrier<-unique(train$CARRIER,incomparables = FALSE)
library(plyr)
a<-count(train, "CARRIER")
Carrier_frequencies<-as.data.frame(a)
library(clusterSim)
Carrier_frequencies$Freq_Normalized <- data.Normalization(a$freq,type = "n1",normalization =
```

- Distance

As the co ordinates was identified for the airports, the distance was calculated using Geospatial packages in R. We thought the distance would be an important factor in trying to identify the ticket fare as it might be linked to the base fare. However, since it was a derived column, it was not completely useful.

MACHINE LEARNING ALGORITHMS

For the mentioned problem statement, we required to implement any predictive machine learning algorithm. We require to predict the “Ticket Fare” for a carrier, on a date, between an origin and destination combination.

This prediction would give the fare based on the features that are selected during the Data preprocess and normalization.

Feature Selection

It is important to select the right features for our prediction. Since the dataset is very huge, it is impossible to keep doing a hit and trial on the dataset. Hence the best method is to, sample the data into n number of rows and try and run any feature selection algorithm on the same.

Test 1:

Features selected – All Features

Algorithm used – Multilinear Regression

Observation:

- It was able to identify the importance of each data.
- It considered all the data as “categorical” data. This would have helped if we had performed 1-N conversion and tried to fit a Neural Network.

(Intercept)	7.248e+02	1.157e+03	0.626	0.53136
X	1.318e-05	1.583e-05	0.832	0.40562
Destin_Value	-1.130e+00	6.579e-01	-1.718	0.08632 .
Origin_Value	-4.725e-01	1.022e+00	-0.462	0.64395
CARRIERAS	1.857e+01	6.567e+00	2.828	0.00485 **
CARRIERB6	3.915e+00	6.798e+00	0.576	0.56487
CARRIERDL	-1.402e+01	8.968e+00	-1.563	0.11862
CARRIERF9	1.395e+01	1.217e+01	1.146	0.25213
CARRIERG4	2.472e+00	2.273e+01	0.109	0.91344
CARRIERNK	1.248e+01	1.684e+01	0.741	0.45915
CARRIERUA	-4.811e+00	3.843e+00	-1.252	0.21115
CARRIERUS	1.275e+01	4.877e+00	2.615	0.00915 **
CARRIERWN	-8.437e+00	5.608e+00	-1.504	0.13300
SEATS	-4.764e-03	3.972e-03	-1.199	0.23082
ORIGINALB	-1.793e-01	9.997e+00	-0.018	0.98569
ORIGINANC	8.311e+00	1.119e+01	0.743	0.45793
ORIGINATL	-7.599e-01	6.381e+00	-0.119	0.90524
ORIGINATW	7.419e+00	1.009e+01	0.735	0.46233
ORIGINAUS	-3.293e+00	7.839e+00	-0.420	0.67458
ORIGINBDL	-9.517e+00	8.512e+00	-1.118	0.26397
ORIGINBHM	-5.671e+00	1.148e+01	-0.494	0.62144
ORIGINBIL	-1.049e+01	1.463e+01	-0.717	0.47382
ORIGINBIS	3.412e+01	2.226e+01	1.532	0.12596
ORIGINBNA	-4.588e+00	7.273e+00	-0.631	0.52838
ORIGINBOS	1.768e+00	7.030e+00	0.251	0.80153
ORIGINBUF	-5.925e+00	8.442e+00	-0.702	0.48305

- The model took only important origin destination combinations, or dates occurring more number of times and treated it as “important”

DEST MSP	8.499e+00	8.623e+00	0.986	0.32473
DEST MSY	1.297e+01	9.226e+00	1.405	0.16045
DEST MTJ	6.057e+01	2.323e+01	2.607	0.00936 **
DEST MWA	7.870e+01	2.390e+01	3.293	0.00105 **
DEST OAK	-1.385e+01	1.368e+01	-1.012	0.31175
DEST OGG	-1.911e+00	1.285e+01	-0.149	0.88187
DEST OKC	1.679e+01	1.105e+01	1.520	0.12915

TEST 2:

Features selected – Ticket Fare, Base Fare, Seats, Origin, Destination, Month of Travel

Observation

- The Destin and the origin were treated as categorical variables.
- Basefare was the only important feature from the test.

```
3 lm.fit <- lm(formula,data = test)
4 formula <- as.formula("TICKET_FARE ~ MONTH_NUMBER+CARRIER+ORIGIN+DEST+BaseFare+SEATS")
5 summary(lm.fit)
6 colnames(test)
7
```

6:1 (Top Level) ↕ R S

Console E:/Boston/NEU/Fall 2016/Additional/Scala/Scala_Project/ ↗

ESTSJC	1.510575	0.254140	0.242	0.00050
ESTSJU	8.361276	7.550443	1.107	0.26834
ESTSLC	7.794365	6.040619	1.290	0.19718
ESTSMF	10.334710	6.522455	1.584	0.11334
ESTSNA	10.881815	8.747593	1.244	0.21375
ESTSRQ	6.989344	8.508313	0.821	0.41154
ESTSTL	10.050809	6.055968	1.660	0.09724
ESTSTT	6.629446	7.938374	0.835	0.40382
ESTSTX	0.691602	9.636185	0.072	0.94280
ESTSWF	8.312469	13.102891	0.634	0.52594
ESTTLH	10.541400	12.304647	0.857	0.39178
ESTTPA	4.434903	6.528319	0.679	0.49706
ESTTRI	NA	NA	NA	NA
ESTTUL	1.792337	7.118103	0.252	0.80124
ESTTUS	10.725264	7.844536	1.367	0.17181
ESTTYS	8.246689	7.549309	1.092	0.27488
aseFare	0.998720	0.006218	160.620	< 2e-16 ***
EATS	-0.002681	0.002682	-1.000	0.31771

--

ignif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

After multiple tests, the following Features were finalized for the data

- TICKET_FARE
- MONTH_NUMBER_OF_TRAVEL
- Carrier_Frequency_Normalized
- Destin_Normalized
- Origin_Normalized
- Base_Fare_Normalized
- Seats_Normalized

Multiple Linear Regression

Multiple linear regression is a method in which we attempt to model a relationship between two or more variables, with a dependent variable for which we will try and fit a linear model. We try and predict for the dependent variable based on “linear relationships” between all the other independent variables.

Mathematically

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \varepsilon_i \text{ for } i = 1, 2, \dots, n.$$

Why Linear Regression?

As per the definition of multi linear regression, every value of the independent value x , depends on the value y . The observed value varies based on all the independent values.

Basefare is one of the most important features which definitely has a relation with the ticketfare. It has been clearly stated that the Ticket price is always greater than the Base fare. Hence it is one of the most important features and it varies linearly, as larger the basefare, larger would be the ticket fare.

For all other features, it had a very good p-value, which allowed those features to be rated as “important”. Since all these features were varying linearly, it was important that we performed prediction using Linear Regression Model.

```

> summary(lm.fit)

Call:
lm(formula = formula, data = train)

Residuals:
    Min       1Q   Median       3Q      Max
-25.1081  -7.9183  -0.7086   6.7256  30.4111

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    132.932751    0.024063  5524.255 < 2e-16 ***
MONTH_NUMBER_OF_TRAVEL -1.512368    0.002576 -587.119 < 2e-16 ***
Carrier_Frequency_Normalized -0.628250    0.009566 -65.678 < 2e-16 ***
Destin_Normalized -0.010285    0.008986  -1.145  0.252
Origin_Normalized -0.046055    0.008985  -5.126 2.97e-07 ***
Base_Fare_Normalized  63.790575    0.009064 7038.172 < 2e-16 ***
Seats_Normalized   -0.007379    0.030715  -0.240  0.810
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.88 on 1499993 degrees of freedom
Multiple R-squared:  0.972,    Adjusted R-squared:  0.972
F-statistic: 8.678e+06 on 6 and 1499993 DF,  p-value: < 2.2e-16

> |

```

As observed from the above screenshot, Features selected are important and have a good p-value.

Running of Linear Regression

R-Script directly allows the regression model to be used using a data and a formula.

```

10 formula <- as.formula("TICKET_FARE ~ MONTH_NUMBER_OF_TRAVEL+Carrier_Frequency_Normalized+Dest:
11 #test_data <- testing[,c("TICKET_FARE", "MONTH_NUMBER", "DAYS", "BaseFare", "Carrier_Frequency_Nor
12 colnames(train)
13 lm.fit <- lm(formula,data = train)
14 modelSummary <- summary(lm.fit)
15 modelSummary$r.squared
16 prediction <- as.data.frame(predict(lm.fit,test))
17 library(data.table)

```

The regression model identifies co-variance between all the features that are fed into the model. The summary of the model will give the p-value.

P-Values are null hypothesis, in which the beta values are set to 1 and how close are they to the predicted model.

LINEAR REGRESSION RESULTS

Summary Result

```

> summary(lm.fit)

Call:
lm(formula = formula, data = train)

Residuals:
    Min       1Q   Median       3Q      Max
-25.1081  -7.9183  -0.7086   6.7256  30.4111

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    132.932751   0.024063  5524.255 < 2e-16 ***
MONTH_NUMBER_OF_TRAVEL -1.512368   0.002576 -587.119 < 2e-16 ***
Carrier_Frequency_Normalized -0.628250   0.009566 -65.678 < 2e-16 ***
Destin_Normalized -0.010285   0.008986  -1.145  0.252
Origin_Normalized -0.046055   0.008985  -5.126 2.97e-07 ***
Base_Fare_Normalized  63.790575   0.009064  7038.172 < 2e-16 ***
Seats_Normalized  -0.007379   0.030715  -0.240  0.810
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.88 on 1499993 degrees of freedom
Multiple R-squared:  0.972,    Adjusted R-squared:  0.972
F-statistic: 8.678e+06 on 6 and 1499993 DF,  p-value: < 2.2e-16

> |

```

Model Coefficients and RMSE value

```

> modelSummary$coefficients

              Estimate Std. Error    t value    Pr(>|t|)
(Intercept)    132.932750794 0.024063470  5524.255368 0.000000e+00
MONTH_NUMBER_OF_TRAVEL -1.512368010 0.002575915 -587.118837 0.000000e+00
Carrier_Frequency_Normalized -0.628250152 0.009565654 -65.677703 0.000000e+00
Destin_Normalized -0.010284960 0.008985516  -1.144615 2.523687e-01
Origin_Normalized -0.046055394 0.008985377  -5.125594 2.966383e-07
Base_Fare_Normalized  63.790574864 0.009063514  7038.172444 0.000000e+00
Seats_Normalized  -0.007378647 0.030714540  -0.240233 8.101496e-01
> modelSummary$r.squared
[1] 0.971998
> |

```

As it can be observed from the screenshot, the RMSE value is 0.97. Ideally the greater the value of the RMSE (>0.70) the better is the prediction. Hence this model has a predicted value that has high accuracy.

Challenges Faced in Linear Regression

- Feature selection

While running the model on the complete dataset, the system was constantly running out of memory. R by default has a memory allocation of your RAM size (8-12GB), and then you can limit the memory size by using the following commands.

```
memory.size(max = FALSE)
memory.limit(size = NA)
```

SOLUTION

Reduced the number of features to predict on the dataset.

- Categorical Variables

All the data including dates, origin, destination was considered as a categorical variable by the model.

SOLUTION

Normalized the data.

NEURAL NETWORK

It is an information processing paradigm, which simulates the working of the “Human Brain”. It consists of a large number of interconnecting nodes which connect with each other and try to solve any complex problem.

As neural networks simulate the working of the human brain, they are the best way to perform any prediction analysis.

Why Neural Network?

Neural network offers a better prediction, as it allows backward propagation. The algorithm repeats a two phase cycle, propagation and weight update.

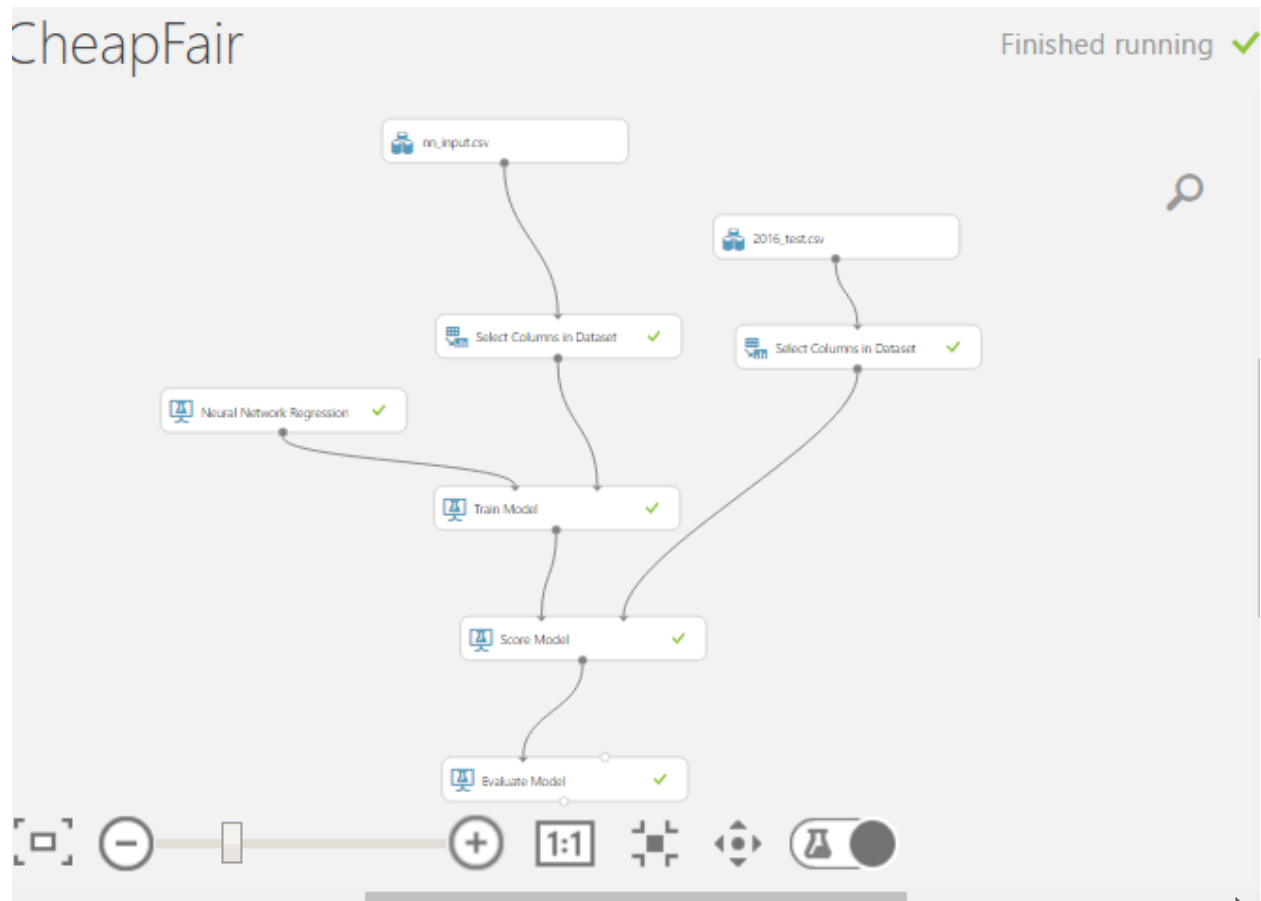
This allows the prediction to be much better and accurate compared to other models.

Neural Network in Azure

The initial plan was to run the neural network in R-Script or Python using Theano and TensorFlow.

However, for this size of the data, every time the system was going out of memory. To avoid this, the setup should be moved to cloud to allow more memory allocation. By moving a code to cloud, allowed us to run it with multiple layers and nodes to get a better accuracy.

Azure is a Microsoft platform, that allows running of Machine learning algorithms on the cloud. We used Azure to run the Neural Network with a hidden layer, to try and predict the results.



Model Setup Steps

- Load the input file to azure storage.
- Get the columns that are required for the model. Since Neural Network requires normalized inputs, we fed in that data to the model.
- Setup the parameters for the training.

Neural Network Regression

Create trainer mode

Single Parameter

Hidden layer specification

Fully-connected case

Number of hidden nodes

100

Learning rate

0.005

Number of learning iterations

100

The initial learning weights diameter

0.1

The momentum

The momentum

0

The type of normalizer

Min-Max normalizer

☒ Shuffle examples

Random number seed

☒ Allow unknown categorical levels

START TIME 12/8/2016 9:49:40 PM

END TIME 12/8/2016 9:49:40 PM





ELAPSED TIME 0:00:00.000

STATUS CODE Finished

STATUS DETAILS Task output was present in output cache

NEURAL NETWORK RESULT

CheapFair > Score Model > Scored dataset

rows	columns			
500000	8			
Carrier_Frequency_Normalized	Origin_Normalized	Destin_Normalized	Scored Labels	
				
2.57216	-1.019468	-0.961955	70.854263	
2.57216	-1.019468	-0.961955	70.837494	
2.57216	-1.019468	-0.961955	70.820694	
2.57216	-1.019468	-0.961955	70.856674	
2.57216	-1.019468	-0.961955	70.856674	
2.57216	-1.019468	-0.961955	70.82193	
2.57216	-1.019468	-0.961955	70.858124	

SUMMARY RESULT

Metrics

Mean Absolute Error	12.948732
Root Mean Squared Error	17.363098
Relative Absolute Error	0.256971
Relative Squared Error	0.071393
Coefficient of Determination	0.928607

Neural Network Challenges

- While feeding all the input columns, the variance was not identified. This caused the resultant variables to be incorrect.

SOLUTION:

Fed in the normalized data from the start, rather than letting the model to normalize.

- Time Taken was too Long.
- Unable to perform bulk loading.

PREDICTIONS

Based on the features that are fed in, and after the algorithm has successfully run, the final values of the prediction are compared. These prediction values are exported as a CSV and fed to the input of a Big Database.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Predicted_Ticket_Fare	Destin_Va	Origin_Va	CARRIER	SEATS	ORIGIN	DEST	MONTH_OF_TRAVEL	MONTH_N	DAY_OF_T	DATE_OF_TRAVEL	DATE_OF_BOOKING	BaseFare	DAY_OF_E	MONTH_N	MONTH_OF
2	85.64738118	1	3	DL		1	ATL	FLL	March	3	14	3/14/2015	11/17/2014	63.50977	17	11 November
3	88.66908741	1	3	DL		4	ATL	FLL	January	1	28	1/28/2015	11/22/2014	63.50977	22	11 November
4	87.15816145	1	3	DL		5	ATL	FLL	February	2	17	2/17/2015	11/28/2014	63.50977	28	11 November
5	87.15813231	1	3	DL		6	ATL	FLL	February	2	15	2/15/2015	11/4/2014	63.50977	4	11 November
6	85.64720634	1	3	DL		7	ATL	FLL	March	3	30	3/30/2015	12/19/2014	63.50977	19	12 December
7	87.15807403	1	3	DL		8	ATL	FLL	February	2	16	2/16/2015	11/18/2014	63.50977	18	11 November
8	87.15804489	1	3	DL		9	ATL	FLL	February	2	16	2/16/2015	2/12/2015	63.50977	12	2 February
9	85.64711892	1	3	DL		10	ATL	FLL	March	3	23	3/23/2015	2/17/2015	63.50977	17	2 February
10	85.64708979	1	3	DL		11	ATL	FLL	March	3	28	3/28/2015	3/25/2015	63.50977	25	3 March
11	88.66882516	1	3	DL		13	ATL	FLL	January	1	21	1/21/2015	11/16/2014	63.50977	16	11 November
12	85.64703151	1	3	DL		13	ATL	FLL	March	3	1	3/1/2015	1/24/2015	63.50977	24	1 January
13	85.64703151	1	3	DL		13	ATL	FLL	March	3	30	3/30/2015	12/14/2014	63.50977	14	12 December
14	87.15781178	1	3	DL		17	ATL	FLL	February	2	7	2/7/2015	1/1/2015	63.50977	1	1 January
15	88.66908741	1	3	DL		17	ATL	FLL	January	1	28	1/28/2015	11/22/2014	63.50977	22	11 November

HBASE DATABASE SETUP

For the data that has been generated, we need to push it to a Big Database setup. For that, we chose HBASE as our database to be used.

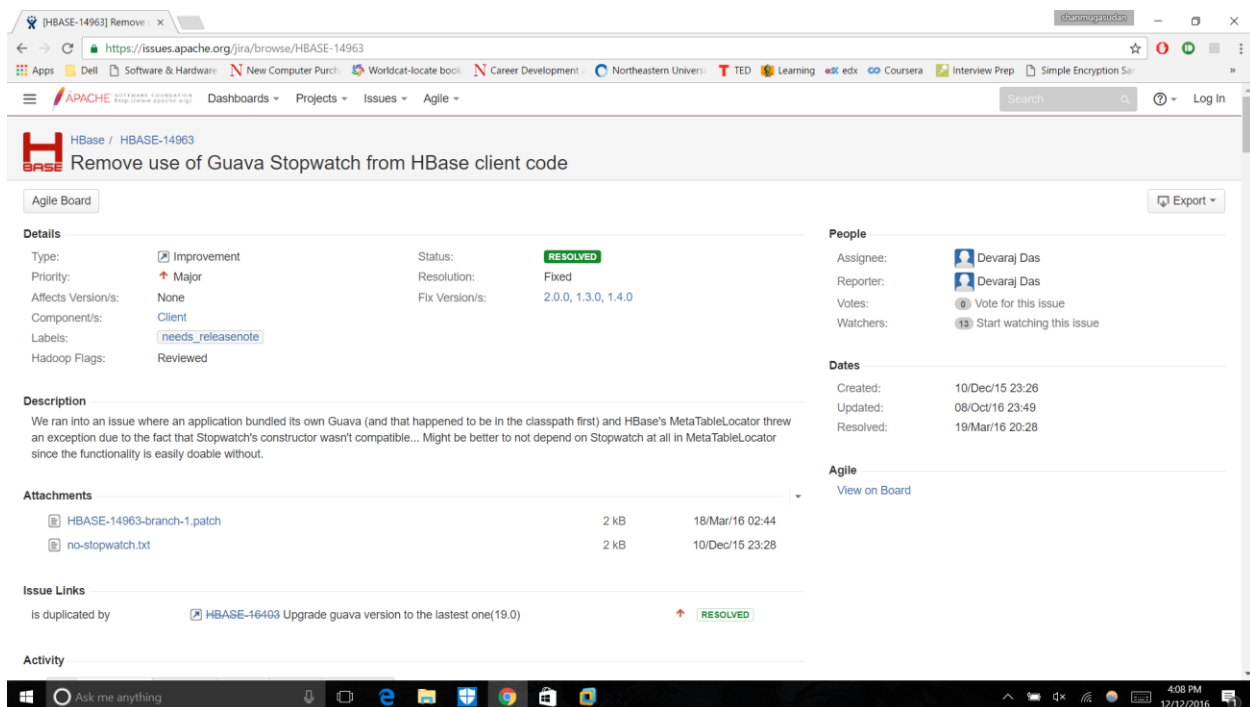
Why HBASE?

HDFS supports sequential access of data. HBASE allows us to perform Random Access of data. HBASE being a columnary data store, helps us in retrieving the data in a quicker way.

HBASE deals with big data effectively, compared to conventional RDBMS. Its distributed architecture helps us to scale it effectively.

Challenges Faced in HBASE

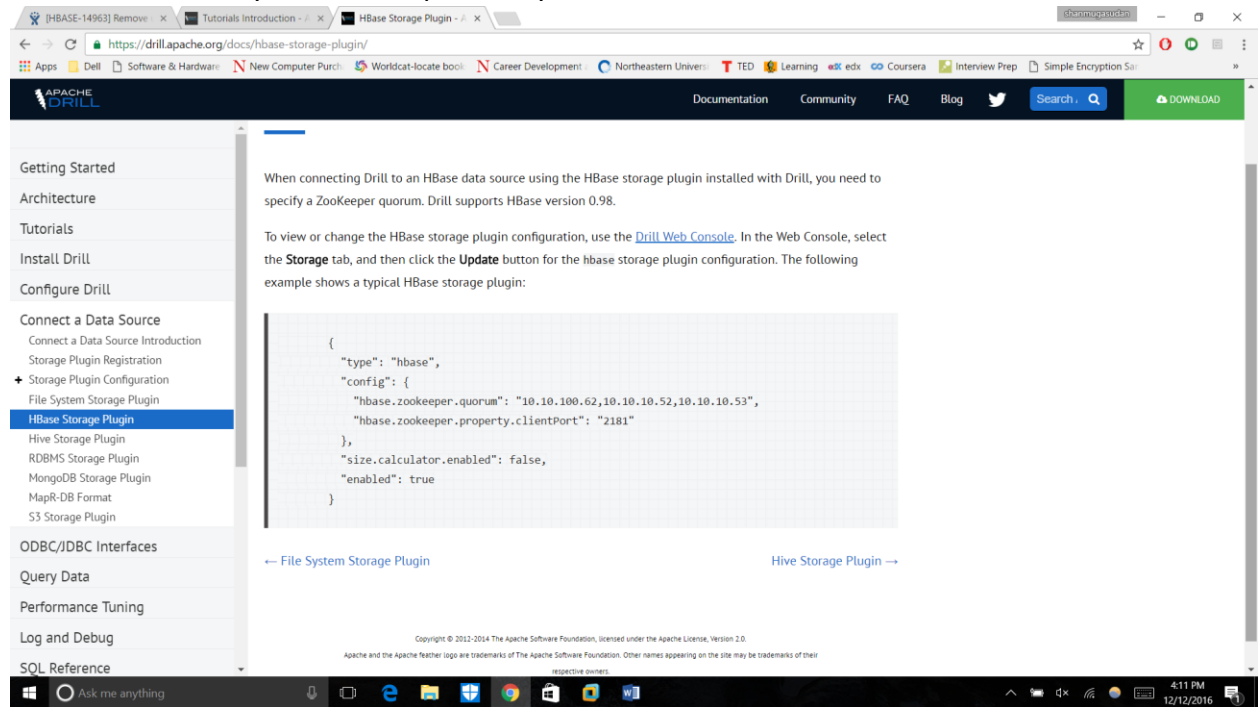
- There is a bug in one of the guava jars(guava.12.0.1) in hbase that prevents accessing data stored on different regions that could be connected to HDFS. We figured out that the hbase binary version can not support the requirement and so decided to work on the source file.
- We cloned the source code via git and used maven to build the code after applying the patch as shown in the below screenshot.
- However, the build took us very long time and so we weren't able to verify the test case.



- Connecting drill via the console

Since the idea was to utilize the REST interface of Drill to integrate with the web UI, we worked more on setting up the drill.

Drill has few connection dependencies that prevented from opening the shell prompt, we had to manually override dependency.



- Querying on Hbase: Drill provides a variety of options to present the retrieved data.
- We decided to go with the SQL format and so it was simple. However, setting up connection strings for Drill web console was tricky.

- We faced challenges in identifying the connection errors due to zookeeper and Drill interface connections.

The query returns results that are not useable. In the next step, you convert the data from byte arrays to UTF8 types that are meaningful.

row_key	account	address
[B@e6d9eb7	{ "name": "QoxpY2U=" }	{ "state": "Q0E=", "street": "MTIZIEJhbGxtZXIgcXV=", "zipco
[B@2823a2b4	{ "name": "Qm9l" }	{ "state": "Q0E=", "street": "MSBjbmZpbm10ZSBmb29w=", "zipco
[B@3b8ec02	{ "name": "RnJhbms=" }	{ "state": "Q0E=", "street": "NDM1IFdhbGt1ciB0dA=", "zipco
[B@242895da	{ "name": "TWFyeQ==" }	{ "state": "Q0E=", "street": "NTYgU291dGhlcm4gUGt3eQ==", "z

4 rows selected (1.335 seconds)

3. Issue the following query, that includes the CONVERT_FROM function, to convert the students table to typed data:

```
SELECT CONVERT_FROM(row_key, 'UTF8') AS studentid,
       CONVERT_FROM(students.account.name, 'UTF8') AS name,
       CONVERT_FROM(students.address.state, 'UTF8') AS state,
       CONVERT_FROM(students.address.street, 'UTF8') AS street,
       CONVERT_FROM(students.address.zipcode, 'UTF8') AS zipcode
FROM students;
```

- Connection Loss Exceptions:
- Zookeeper on fully distributed cluster setup was a challenge. We faced connection issues due to incorrect mapping in xml, in correct IP address and zookeeper config files.

asked May 27 '11 at 14:18
Charles5
501 ● 1 ● 6 ● 24

3 Answers

Charles,
7
This is a Zookeeper(ZK) error. The HBase client tries to get the /hbase node from Zookeeper and fails.
You can get a ZK dump from the HBase master web interface. You should see all the connections to ZK and figure out if something is exhausting them.
Before diving into anything else you could try restarting your ZK cluster and see if it fixes your problem. (It's strange that you see that with a single client).
HBase has a setting to increase the number of connections to ZK. It's
`hbase.zookeeper.property.maxClientCnns`
There were a few updates (see below) lately related to the default number of connections (there's a hbase-default.xml file that has all the default configurations). You can override this in your hbase-site.xml file (under HBase conf dir) and raise it to 100 or more. But make sure you're not masking the real problem this way, you shouldn't see this problem with a single client.
We've had a similar situation, but it was happening during heavy operations from map-reduce jobs, after upgrading to HBase-0.90.
Here are a couple of issue related to your problem:

- <https://issues.apache.org/jira/browse/HBASE-3773>
- <https://issues.apache.org/jira/browse/HBASE-3777>

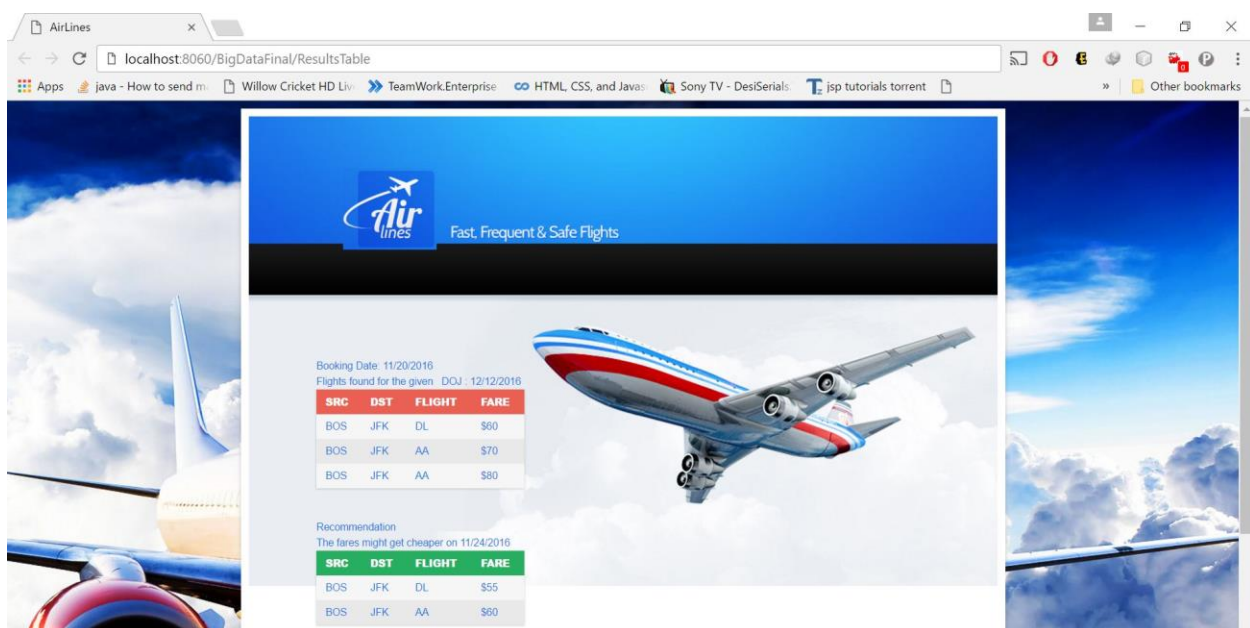
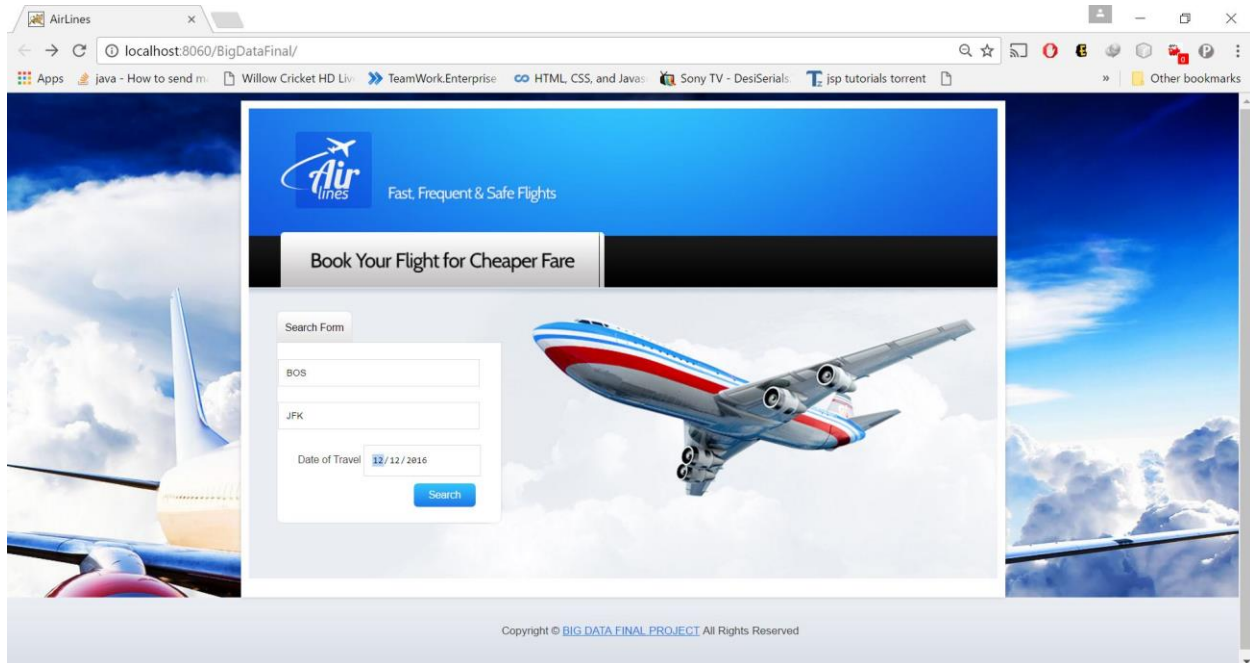
If you still can't figure it out send an email to the hbase-users list or join the #hbase channel on freenode and ask live questions.

answered May 30 '11 at 9:03
Cosmin Lehene
246 ● 2 ● 6

HBASE connection using JAVA

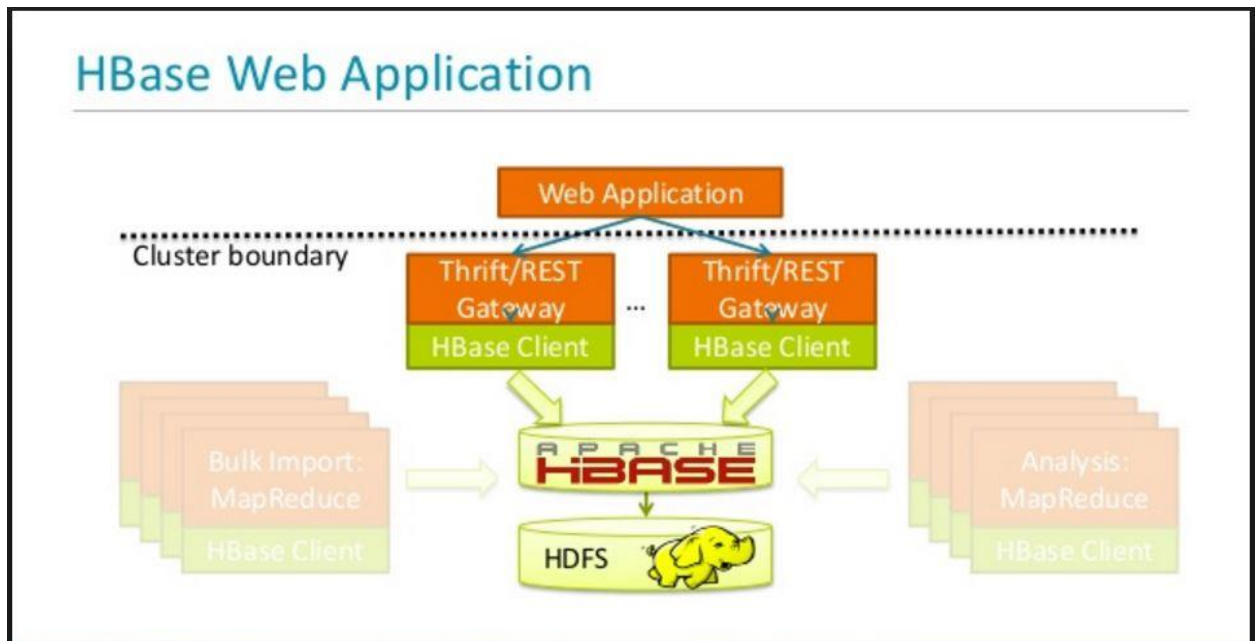
USER INTERFACE

A user interface was designed to integrate the result and push it to an output for the UI. We used a simple MVC for the UI.



CHALLENGES FACED IN UI

- The user interface was not very easy to integrate with HBASE. This is because as per the architecture, there needs to be another interface between the big data base and the User Interface.



FUTURE SCOPE

This project can help a number of customers to benefit their ticket bookings, especially during holiday season. Here are some of the challenges that are there in terms of the project scope

- Airlines do not want to disclose their prices to their competitors.
- It is very difficult to monitor the fluctuating factors.
Eg: Even if the price of the ticket is cheap on end of December, if the oil prices increase due to the victory of Donald Trump, the prices would definitely increase as well.

Hence, after looking at these challenges, it is important that this idea would grow, only after all airlines are aligned and agree to a common platform by sharing their data. We can extend the scope of the project by

- Integrating and scraping data from each airline to get real prices of their flights between the source and destination combination
- Integrating the exchange rates and oil prices into consideration at the dataset level.
- Using various Business Intelligence techniques to try and find a pattern for the data.
- Increasing the training sample for the model for each year.

BIBLIOGRAPHY

Official Stanford Paper for the idea

<http://cs229.stanford.edu/proj2012/Papadakis-PredictingAirfarePrices.pdf>

References

<http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm>

<https://en.wikipedia.org/wiki/Backpropagation>

https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

