CS439 Course Project

Predicting Future Airfare

Da Huo Yijia Chen

April 29, 2018

1. Introduction

In the current age, the number of passengers carried by airplanes is growing rapidly, and pricing is a major concern for both the airlines and customers.

The proposed research is aimed at studying the principal patterns of flight ticket prices from various airlines. After we find and visualize the patterns, we will be able to compare the pricing for the certain routes in the different time periods or from the different airlines in order to predict the best deal for the certain travel. We will also discover three seasonal indicators (high season, shoulder season and low season) since usually people try to avoid the peak season to get the low cost deal. And high season is characterized by the highest ticket prices, shoulder season by average prices, and low season by the lowest prices.

Specifically, we will use advanced programming language to parse the keywords and extract data from the website, visualize the collected information and predict the future pricing patterns using statistical tools. For this intermediate report, we use $EWR \rightarrow PEK$ as the initial example.

2. Data Collection

2.1 Web Crawling (Python)

Since out team is doing flight ticket price analyzing, So we need to perform web scraping on major airlines or third party ticket selling website. An very important action of scarping flight ticket is to complete the search form which provides the depart location, arrive location, depart date, arrive date, etc. This is a very complex task to complete given that most website does not following a standard form HTTP Request but using JavaScript to make HTTP request instead, and most airliner's websites are using anti-crawling software.

2.1.1 United Airlines

We first tried to use the Python web library Mechanize, which is a tool to help us fake a form submission, to perform web scraping on United Airlines' website. We were able to find out the requirement of each search form control input and filling out the form correctly. However, we were being detected by the browser as a robot and get the HTTP response 403 Forbidden. To resolve that, we have fake an user agent and pretend we were a real user agent. However for some reason, United Airlines' website is not responding to the form submission. We tried the same code on Facebook.com on the login form and it works perfectly. So we assume UA is using some anti-crawling strategy and prevent us from accessing.

2.1.2 Air China

After attempting with UA.com, we switch to scrap the website of Air China, However, after spending some time inspecting their code, we found that Air China are mostly using JavaScript to dynamically generate the HTTP Request. All of their input field are represented by div tags and span tags and there were no form tag at all. This makes it very hard to use Mechanize to fake a user form submission given that Mechanize relies heavily on the function br.forms() to find the form tags inside the HTML. Therefore, it is not possible to get information from Air China using Mechanize.

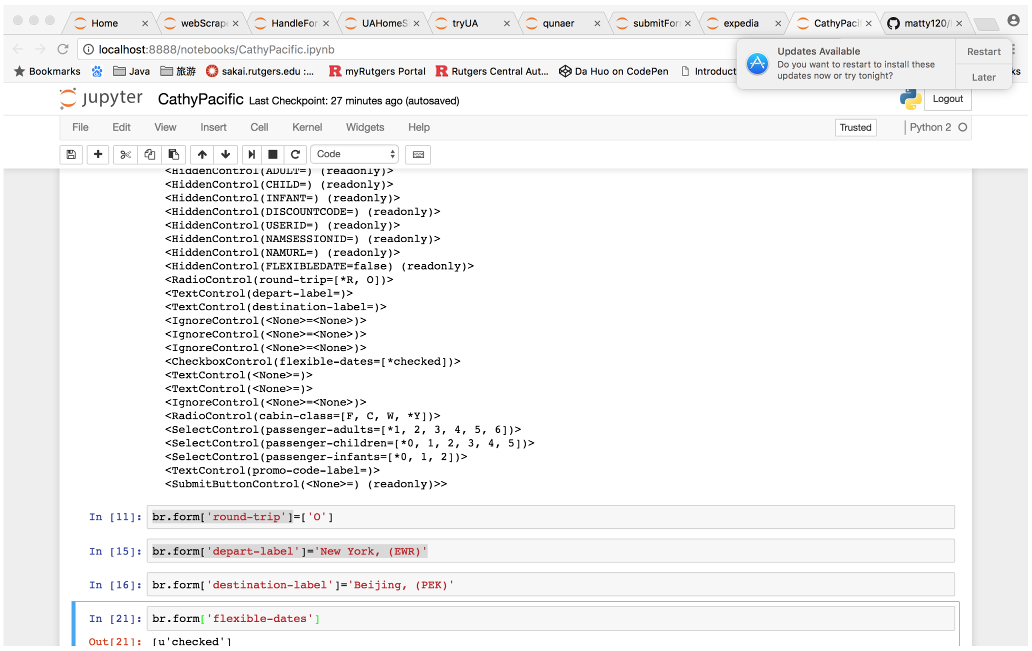
2.1.3 Cathy Pacific

Then we switch to Cathy Pacific's website. This site is the one which we put most of our effort to besides the one we successfully scraped. Cathy Pacific's search form from there website is fairly simple but with some traps. For example, the actual text inputs and select inputs are not relevant to the data submitted to their server at all, after a few failures on submitting the form, we found that the only form control that matters in their website are the read only hidden controls as the figure 1 shows

These controls are automatically filled by JavaScript based on the value and this is the reason of why we were keep failing on the form submission. To resolve that, we made a actual form submission through the real web browser and captures the following as figure 2 shows:

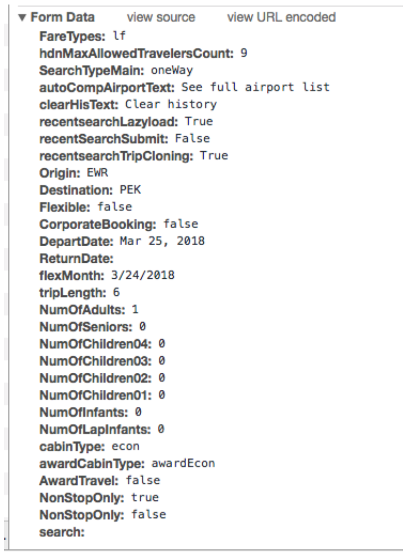
So we were able to fake a form by following the real form submitted through Chrome. However, the response we got back is just a intermediate page which use JavaScript to auto-redirect to the real Search Result as figure 3 shows.

This makes the problem get a lot more complex since the Mechanize Browser object cannot handle any JavaScript . After a few days of struggling, we figured out that the only way to do this is through the real browser. And we were able to find the web driver tool Selenium to help us to do this.



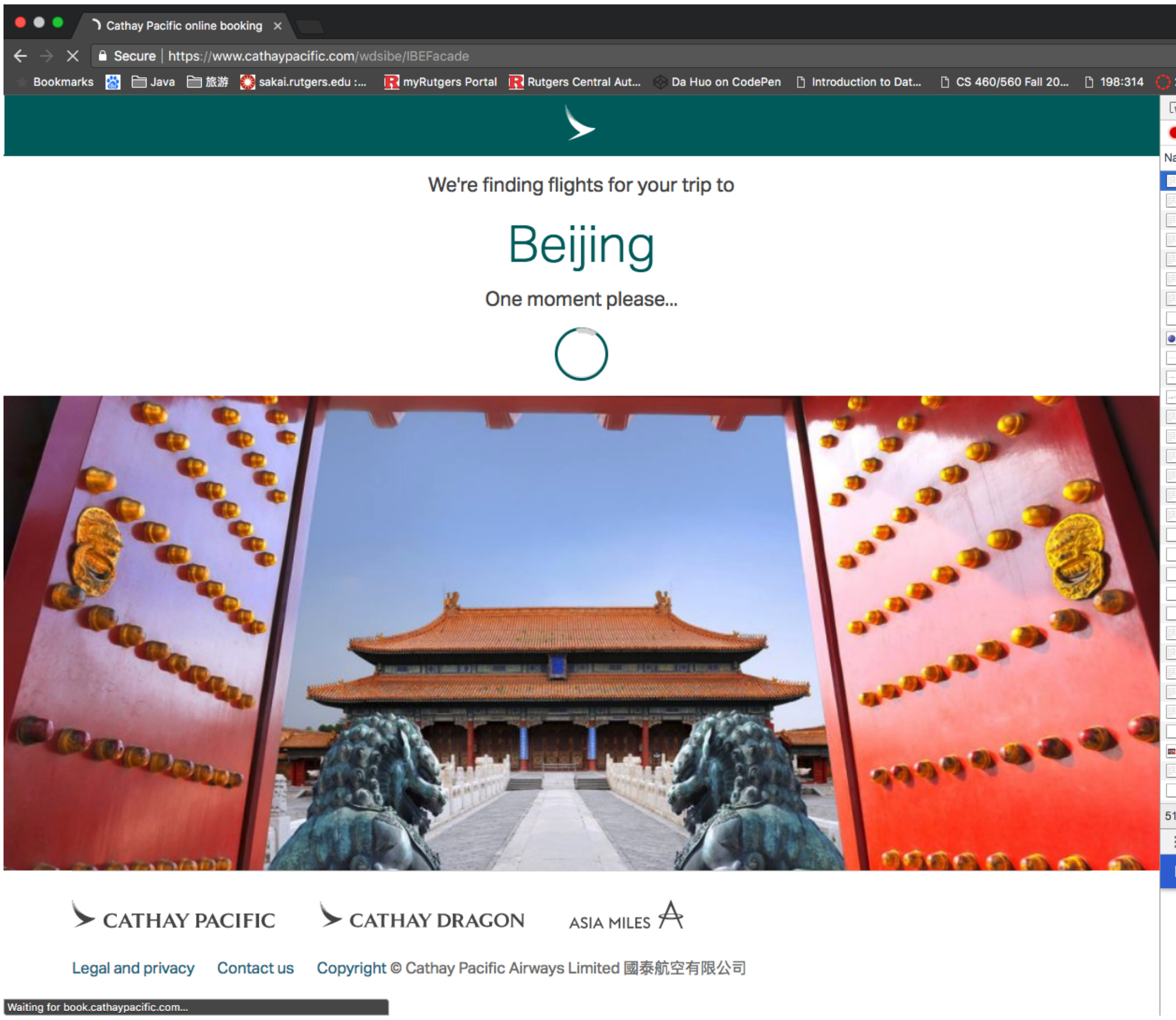
```
            <HiddenControl(ADULT=) (readonly)>
            <HiddenControl(CHILD=) (readonly)>
            <HiddenControl(INFANT=) (readonly)>
            <HiddenControl(DISCOUNTCODE=) (readonly)>
            <HiddenControl(USERID=) (readonly)>
            <HiddenControl(NAMSESSIONID=) (readonly)>
            <HiddenControl(NAMURL=) (readonly)>
            <HiddenControl(FLEXIBLEDATE=false) (readonly)>
            <RadioControl(round-trip=[*R, O])>
            <TextControl(depart-label=)>
            <TextControl(destination-label=)>
            <IgnoreControl(<None>=<None>)>
            <IgnoreControl(<None>=<None>)>
            <IgnoreControl(<None>=<None>)>
            <CheckboxControl(flexible-dates=[*checked])>
            <TextControl(<None>=)>
            <TextControl(<None>=)>
            <IgnoreControl(<None>=<None>)>
            <RadioControl(cabin-class=[F, C, W, *Y])>
            <SelectControl(passenger-adults=[*1, 2, 3, 4, 5, 6])>
            <SelectControl(passenger-children=[*0, 1, 2, 3, 4, 5])>
            <SelectControl(passenger-infants=[*0, 1, 2])>
            <TextControl(promo-code-label=)>
            <SubmitButtonControl(<None>=) (readonly)>>

In [11]: br.form['round-trip']=['O']

In [15]: br.form['depart-label']='New York, (EWR)'

In [16]: br.form['destination-label']='Beijing, (PEK)'

In [21]: br.form['flexible-dates']
Out[21]: [u'checked']
```

Figure 1: Controls in Cathay Pacific Search Form



```
FareTypes: lf
hdnMaxAllowedTravelersCount: 9
SearchTypeMain: oneWay
autoCompAirportText: See full airport list
clearHisText: Clear history
recentsearchLazyload: True
recentSearchSubmit: False
recentsearchTripCloning: True
Origin: EWR
Destination: PEK
Flexible: false
CorporateBooking: false
DepartDate: Mar 25, 2018
ReturnDate:
flexMonth: 3/24/2018
tripLength: 6
NumOfAdults: 1
NumOfSeniors: 0
NumOfChildren04: 0
NumOfChildren03: 0
NumOfChildren02: 0
NumOfChildren01: 0
NumOfInfants: 0
NumOfLapInfants: 0
cabinType: econ
awardCabinType: awardEcon
AwardTravel: false
NonStopOnly: true
NonStopOnly: false
search:
```

Figure 2: Intercepted Form Data



We're finding flights for your trip to

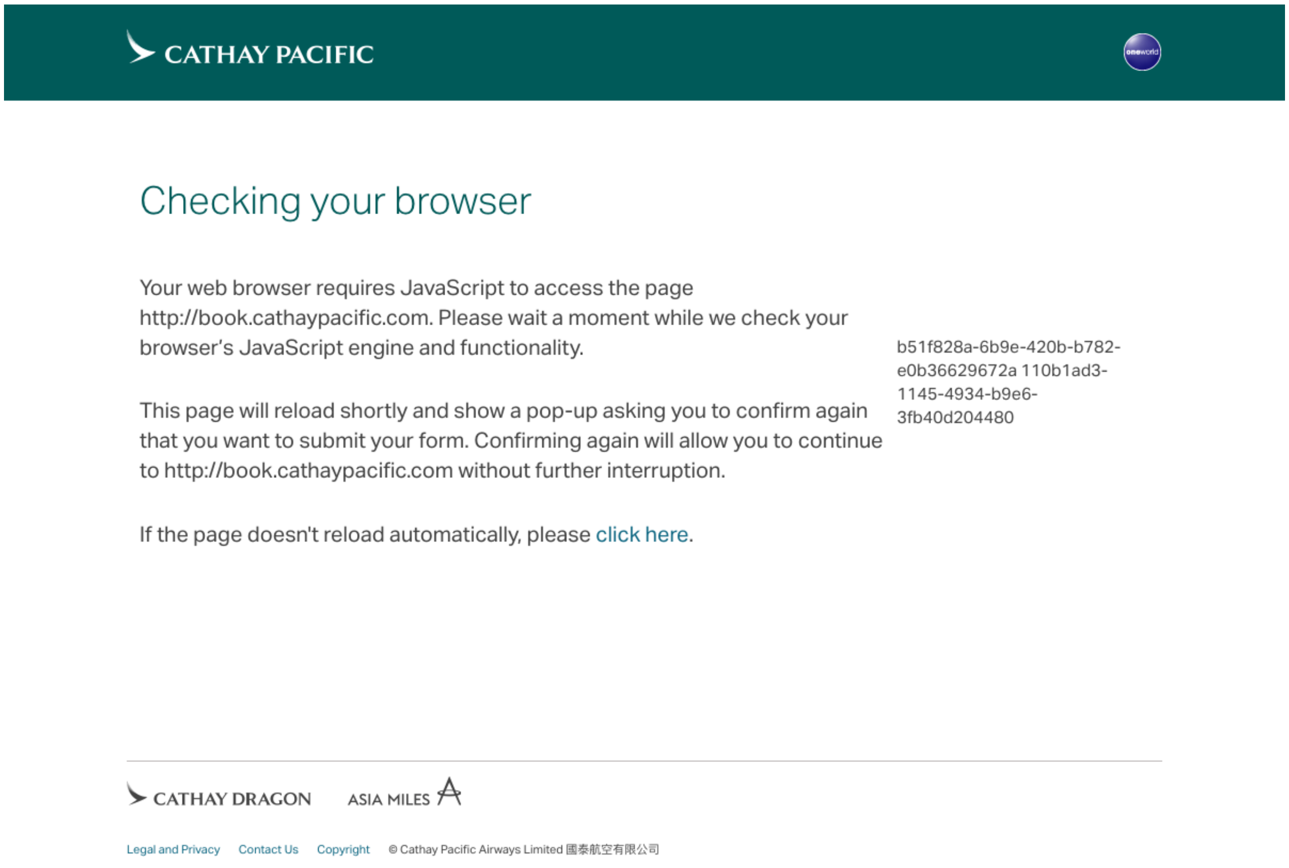
Beijing

One moment please...

CATHAY PACIFIC
CATHAY DRAGON
ASIA MILES
Legal and privacy
Contact us
Copyright © Cathay Pacific Airways Limited 國泰航空有限公司
Waiting for book.cathaypacific.com...

Figure 3: Intermediate Page in Cathay Pacific Website

However, for some reason, when we use Selenium to open this response page, the JavaScript gets blocked and we were not able to view the search result as figure 4 shows:


CATHAY PACIFIC

Checking your browser

Your web browser requires JavaScript to access the page http://book.cathaypacific.com. Please wait a moment while we check your browser's JavaScript engine and functionality.

b51f828a-6b9e-420b-b782-e0b36629672a 110b1ad3-1145-4934-b9e6-3fb40d204480

This page will reload shortly and show a pop-up asking you to confirm again that you want to submit your form. Confirming again will allow you to continue to http://book.cathaypacific.com without further interruption.

If the page doesn't reload automatically, please click here.

CATHAY DRAGON ASIA MILES

Legal and Privacy Contact Us Copyright © Cathay Pacific Airways Limited 國泰航空有限公司

Figure 4: Selenium Been Blocked

It turns out to be that Cathy Pacific is using the anti-crawling software from Distil Networks which blocks all connection from Selenium. Therefore, we failed to get information from Cathy Pacific.

2.1.4 C-Trip

Finally, we targeted the Chinese third party flight ticket selling website, CTrip.com. This website seems to have no protection on anti-robot or anti-crawling. We use Selenium web driver to locate the Origin location, Destination location, etc. and we can use the `Send_Keys()` function to directly send text input as if we were typing on the keyboard. However, we still face a few challenges. The first challenge is that, this website only accept the Origin and Destination from their pop up window. This window is dynamically generated by JavaScript and is hard to locate using software. The solution we found is that to first give some text input into the Origin input text box. Then we use time.sleep(1) to wait for 1 second until the popup windows appears. Then we use `get_element_by_id()` to locate this window and use the click() function to select it.

The second challenge is that, to select a depart date that is beyond April, we need to click on the next page button several times. How many times do we click on it is different from search to search. So we kept a month variable which represent current month and use the month we want to search on to find the difference, and then click the next page n times.

After we got the response search result, we use the BeautifulSoup library to locate the correspond tags inside the HTML which contains the information we want. Then we can use the `tag.get_text()` function to get the information we need.

2.2 Data Format

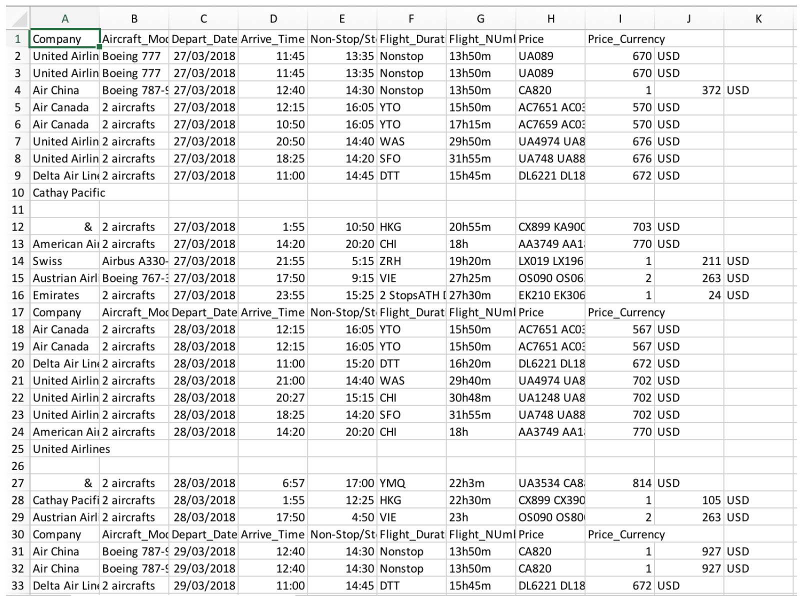
We store our datasets in a python dictionary, a mutable mapping of unique keys to values. We set company, date, price, and duration as our keys with the corresponding values. Then the program will write those data to a CSV file since writing and reading from CSV are generally easier.

2.3 Data Preprocessing

We choose to save our data as an CSV file. When we first got the data back, the date is not well formatted and with multiple problems as figure 5 shows:

Two major reasons that cause these problems are:

1. The ticket price we get back from the browser contains ","s such as : "$5,345". This comma makes the whole document not well formatted and is hard to process



	A	B	C	D	E	F	G	H	I	J	K
1	Company	Aircraft_Moc	Depart_Date	Arrive_Time	Non-Stop/St	Flight_Durat	Flight_NUml	Price	Price_Currency		
2	United Airlin	Boeing 777	27/03/2018	11:45	13:35	Nonstop	13h50m	UA089	670	USD	
3	United Airlin	Boeing 777	27/03/2018	11:45	13:35	Nonstop	13h50m	UA089	670	USD	
4	Air China	Boeing 787-9	27/03/2018	12:40	14:30	Nonstop	13h50m	CA820	1	372	USD
5	Air Canada	2 aircrafts	27/03/2018	12:15	16:05	YTO	15h50m	AC7651 AC03	570	USD	
6	Air Canada	2 aircrafts	27/03/2018	10:50	16:05	YTO	17h15m	AC7659 AC03	570	USD	
7	United Airlin	2 aircrafts	27/03/2018	20:50	14:40	WAS	29h50m	UA4974 UA8	676	USD	
8	United Airlin	2 aircrafts	27/03/2018	18:25	14:20	SFO	31h55m	UA748 UA88	676	USD	
9	Delta Air Line	2 aircrafts	27/03/2018	11:00	14:45	DTT	15h45m	DL6221 DL18	672	USD	
10	Cathay Pacific										
11											
12	&	2 aircrafts	27/03/2018	1:55	10:50	HKG	20h55m	CX899 KA900	703	USD	
13	American Air	2 aircrafts	27/03/2018	14:20	20:20	CHI	18h	AA3749 AA1	770	USD	
14	Swiss	Airbus A330-	27/03/2018	21:55	5:15	ZRH	19h20m	LX019 LX196	1	211	USD
15	Austrian Airl	Boeing 767-3	27/03/2018	17:50	9:15	VIE	27h25m	OS090 OS06	2	263	USD
16	Emirates	2 aircrafts	27/03/2018	23:55	15:25	2 StopsATH [	27h30m	EK210 EK306	1	24	USD
17	Company	Aircraft_Moc	Depart_Date	Arrive_Time	Non-Stop/St	Flight_Durat	Flight_NUml	Price	Price_Currency		
18	Air Canada	2 aircrafts	28/03/2018	12:15	16:05	YTO	15h50m	AC7651 AC03	567	USD	
19	Air Canada	2 aircrafts	28/03/2018	12:15	16:05	YTO	15h50m	AC7651 AC03	567	USD	
20	Delta Air Line	2 aircrafts	28/03/2018	11:00	15:20	DTT	16h20m	DL6221 DL18	672	USD	
21	United Airlin	2 aircrafts	28/03/2018	21:00	14:40	WAS	29h40m	UA4974 UA8	702	USD	
22	United Airlin	2 aircrafts	28/03/2018	20:27	15:15	CHI	30h48m	UA1248 UA8	702	USD	
23	United Airlin	2 aircrafts	28/03/2018	18:25	14:20	SFO	31h55m	UA748 UA88	702	USD	
24	American Air	2 aircrafts	28/03/2018	14:20	20:20	CHI	18h	AA3749 AA1	770	USD	
25	United Airlines										
26											
27	&	2 aircrafts	28/03/2018	6:57	17:00	YMQ	22h3m	UA3534 CA8	814	USD	
28	Cathay Pacifi	2 aircrafts	28/03/2018	1:55	12:25	HKG	22h30m	CX899 CX390	1	105	USD
29	Austrian Airl	2 aircrafts	28/03/2018	17:50	4:50	VIE	23h	OS090 OS80	2	263	USD
30	Company	Aircraft_Moc	Depart_Date	Arrive_Time	Non-Stop/St	Flight_Durat	Flight_NUml	Price	Price_Currency		
31	Air China	Boeing 787-9	29/03/2018	12:40	14:30	Nonstop	13h50m	CA820	1	927	USD
32	Air China	Boeing 787-9	29/03/2018	12:40	14:30	Nonstop	13h50m	CA820	1	927	USD
33	Delta Air Line	2 aircrafts	29/03/2018	11:00	14:45	DTT	15h45m	DL6221 DL18	672	USD	

Figure 5: Raw Data Before Preprocessing

2. The airline name we get back from browser contains multiple white spaces and special characters.

To resolve that, we pre-processed the data before we save them into an CSV file. First, we take each ticket price and search for "," inside it. If true, then we delete that comma. Second, we found a pattern on how the company contains special characters: when there are multiple airlines perform the same route, CTrip uses an icon to represend the second airliner. So as a result , we always get special characters. The solution is that we first search for flight which contains more than one airliner, and then we process the String to get rid of the white spaces and special characters and leave only the letters.

Another part which requires data pre-processing is that to make our data suitable for machine learning algorithms. Since most machine algorithms requires numerical features, so we have to convert our features such as departure date to a numerical form. We wrote a python program to pre-process our read-in data. We performed the following changes to our data:

- The departure date from "2018-03-24" to an integer format 20180324
- We created a table map different Airlines to a unique code, for example. use 0 to represent United Airlines.

- We change Non-stop only to a boolean format, use 1 for non-stop flights, use 0 for other flights.
- We change the duration of flight, from the format x hours y minutes to minute-unite only.

After the above changes, we were safely and successfully performed our machine learning experiment.

3. Data Analysis and Visualization

We take Air China One-Way Flight from EWR to PEK as an example, and calculate the average weekly prices from 2018-4-21-2019-4-19 (52 weeks). We use R, Python library and multiple data visualization tools to create the following charts as figure 6, figure 7, and figure 8 shows:

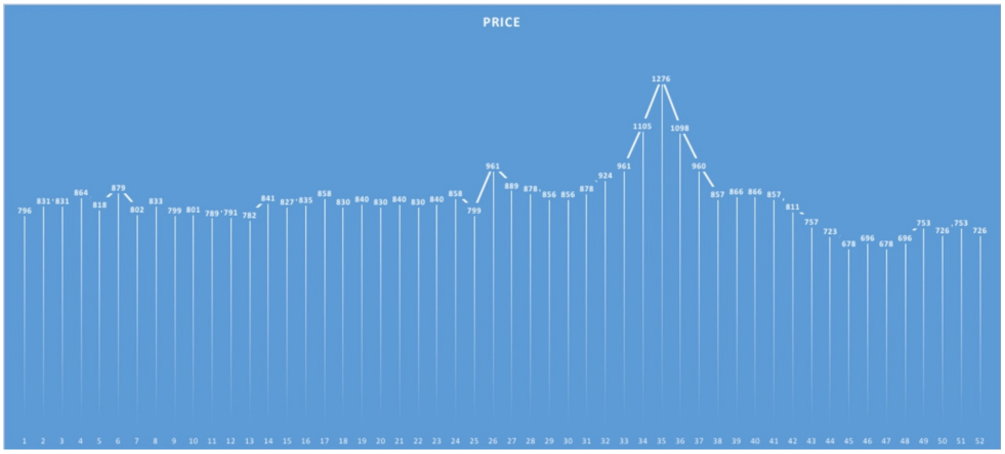


Figure 6: Visualization Graph of Data

From these graphs, we can see that the high season is from November to January and the peak time is Christmas week. That is because a lot of students will go back to China during the winter holiday. February - April should be the low season since the prices in this period is extremely low, and another reason for this situation is that this time period is still far away.

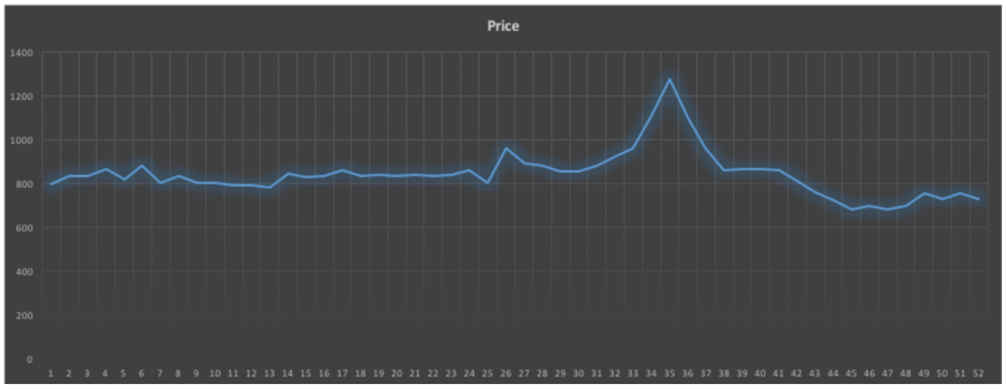


Figure 7: Visualization Graph of Data

4. Machine Learning and Prediction

The prediction of future airfare generally can be viewed as both a regression problem and a classification problem. If we view this problem as a regression problem, then our goal is to predict a specific price given a set of features. If we view this problem as a classification problem, then our goal is to predict a general class of a given a set of features, for example, \$0 – \$499 or \$500 – \$999

We performed 4 machine learning experiment with 4 algorithms. We performed KNN regression algorithm for regression problem, and we used KNN classification, perceptron, and naive bayes for classification problem.

4.1 KNN Regression

4.1.1 Features

In this machine learning experiment, we used the following features:

- Airline name
- Departure date
- Duration of flight
- Non-stop only or not

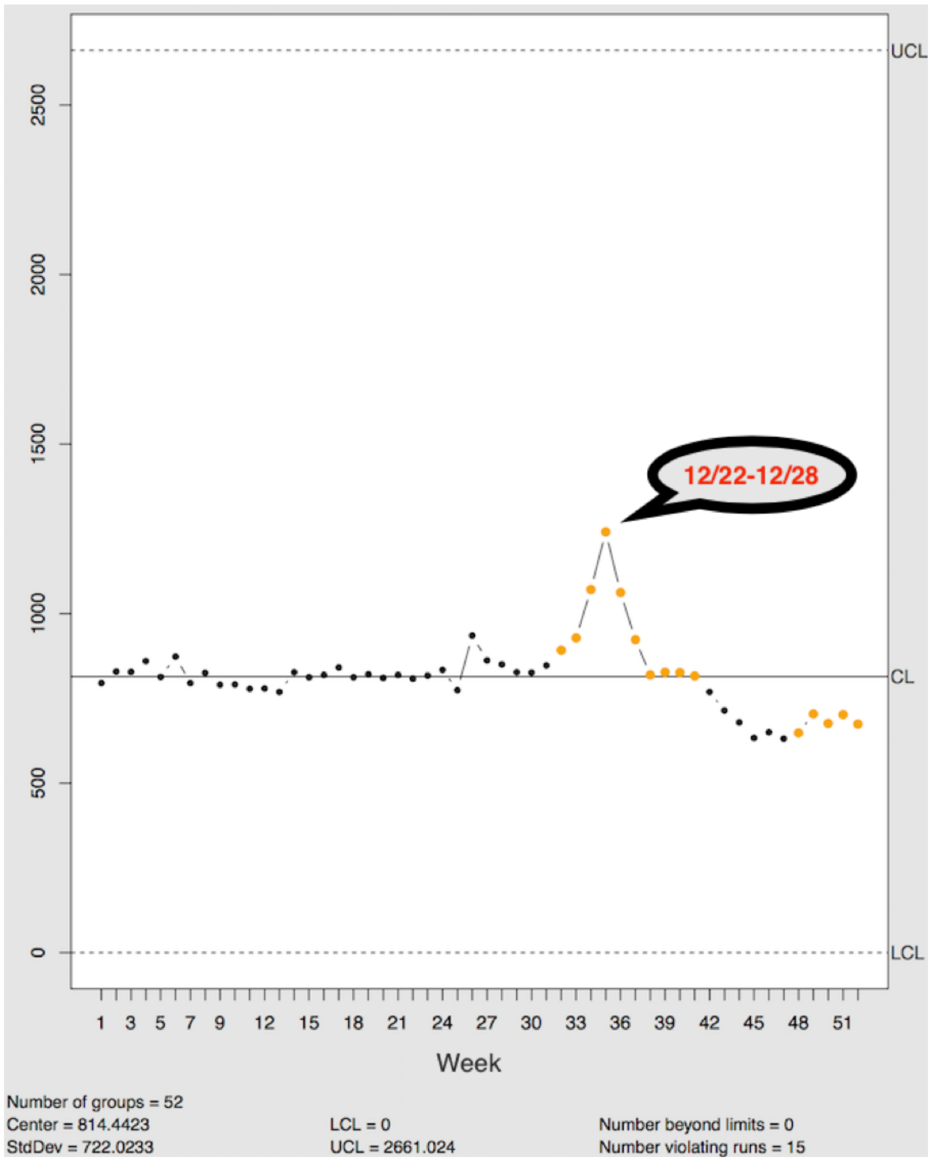


Figure 8: Visualization Graph of Data

4.1.2 Experiment Results

After performing KNN regression on our dataset, we get the following average explained variance score on different size of neighbors as well as different cross validation mechanism:

- 40 Neighbors with 20-Fold cross validation : 0.0834657606956
- 40 Neighbors with 10-Fold cross validation : 0.095068233224
- 20 Neighbors with 20-Fold cross validation : 0.225607445789
- 20 Neighbors with 10-Fold cross validation : 0.236448776231

In this experiment, 40 neighbors with 20-fold cross validation gives the best result with a variance ascore of 0.08. The prediction results are generally following the trend of the actual result. We generate a graph comparing prediction and actual results as figure 9 shows:

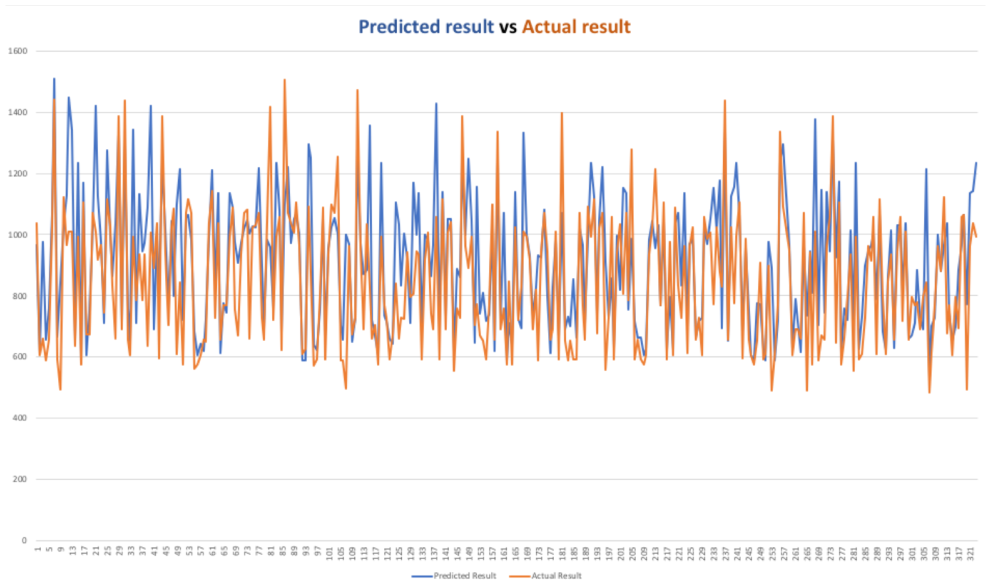


Figure 9: Predicted Results vs. Actual Results

4.2 Perceptron Classifier

4.2.1 Features

In this machine learning experiment, we used the following features:

- Airline name
- Departure date
- Duration of flight
- Non-stop only or not

4.2.2 Experiment Results

In this experiment, we break the prediction results to 5 classes:

- \$0 – \$499
- \$500 – \$999
- \$1000 – \$1499
- \$1500 – \$1999
- \$2000 or above

After performing Perceptron Classifier with max iteration = 10000 on our dataset, we get the following accuracy score compare to the actual result on different cross validation mechanism:

- Perceptron with 10-Fold cross validation: $0.449591280654 \approx 45\%$
- Perceptron with 20-Fold cross validation : $0.601092896175 \approx 60\%$
- Perceptron with 40-Fold cross validation : $0.681318681319 \approx 68\%$

In this experiment, although the accuracy of prediction increased as we provide more training data, the prediction results is still not good enough given that we only have limited training data.

4.3 Naive Bayes Classifier

4.3.1 Features

In this machine learning experiment, we used the following features:

- Airline name
- Departure date
- Duration of flight
- Non-stop only or not

4.3.2 Experiment Results

In this experiment, we break the prediction results to 5 classes:

- \$0 – \$499
- \$500 – \$999
- \$1000 – \$1499
- \$1500 – \$1999
- \$2000 or above

We used Gussian Naive Bayes classifier in this experiment. After performing Naive Bayes Classifier on our dataset, we get the following accuracy score compare to the actual result on different cross validation mechanism:

- Naive Bayes with 10-Fold cross validation: 0.677656675749 $\approx$ 67%
- Perceptron with 20-Fold cross validation : 0.663387978142 $\approx$ 66%
- Perceptron with 40-Fold cross validation : 0.681043956044 $\approx$ 68%

In this experiment, the accuracy score does not change too much based on different training data size, it is always around 67%. This experiment result is still not good enough.

4.4 KNN Classifier

4.4.1 Features

In this machine learning experiment, we used the following features:

- Airline name
- Departure date
- Duration of flight
- Non-stop only or not

4.4.2 Experiment Results

After performing KNN Classifier on our dataset, we get the following accuracy score on different size of neighbors as well as different cross validation mechanism:

- 40 Neighbors with 20-Fold cross validation : $0.760109289617 \approx 76\%$
- 40 Neighbors with 10-Fold cross validation : $0.762397820163 \approx 76\%$
- 20 Neighbors with 20-Fold cross validation : $0.800546448087 \approx 80\%$
- 20 Neighbors with 10-Fold cross validation : $0.794005449591 \approx 79\%$

In this experiment, we get the highest accuracy score, 80%, with 20 neighbors and 20-Fold cross validation, which is a good and acceptable prediction accuracy given current training data size and features. But we still think we can improve this accuracy by providing more training data and features.

5. Future Directions

5.1 Future Web Crawling

One possible future direction is to keep getting more data from the web. A major reason of that our machine learning suffering a poor performance is the lack of traning data. Since online ticket price website only provides one year of ticket pricing, so we keep to keep crawling everyday.

5.2 Other Feature to Use

As we get more training data, we could use one more feature, which is days from the search date. The price of a particular flight will change everyday. We need to record its price on every single day in order to perform a more accurate prediction. So we could use days from the search date as another feature if we get enough training data.