When to Book: Predicting Flight Pricing

Qiqi Ren
Stanford University
qiqiren@stanford.edu

Abstract

When is the best time to purchase a flight? Flight prices fluctuate constantly, so purchasing at different times could mean large differences in price. This project uses machine learning classification in order to predict at a given time, considering properties of a flight, whether one should book the flight or wait for a better price.

1 Introduction

As someone who purchases flights frequently, I would like to be able to predict when the best time is to buy in order to get the best deal. I have heard some people claiming that certain days of the week are when prices are lowest, and it's likely that that demand for flights is higher at certain hours, which could indicate higher prices. It's possible this varies depending on various other aspects, like the length of the flight, the date and time of the flight, and how much time there is until the flight.

In this project, I chose to focus on aspects that are visible on the consumer side and predict only the binary class of whether the price will increase or not, which is essentially whether one should buy now or wait. The input is the time of the day of the request, the time of the week of the request, the time of the day of the flight, the time of the week of the flight, the number of stops, the duration of the flight, the number of hours between the request and the flight, and the current price of the flight. The output is the previously described binary class with 1 indicating "should wait" and 0 indicating "should buy."

2 Related Work

There is some existing work in the area of predicting flight prices, though some of the existing services that do this do not publicly share their techniques. Most other authors commented on the two different possibilities of doing this, which were regression in order to predict the value of the price, and binary classification to make conclusions about the price range and whether or not one should buy.

Papadakis [2] used the same approach as I did of binary classification in order to turn the problem into a supervised classification problem, but used a wider range of features including some that I feel may not be information available to consumers, like the number of unsold seats, and features based on historical data, like the recent price for the same ticket, and used the Ripple Down Rule Learner algorithm to model this data.

Similarly, Lu [4] considered historical data features, like the maximum and minimum price so far, and used a neural network as one of their models. Since the neural network achieved poor performance and is computationally expensive, I decided to exclude that model from my own experiments.

One notable commonly cited source in this field is To Buy or Not to Buy: Mining Airfare Data [1] to Minimize Ticket Purchase Price, which looks at how flights change in price by examining price changes on a particular flight number and itinerary. The authors did not use any of the same algorithms as I did, and quantified their results through a simulation in which they predicted total savings from using each algorithm.

The existing work that most aligns with this project is Airfare Prices Prediction Using Machine Learning Techniques [3], which split the project into feature selection and then algorithm selection, and used a similar set of features and algorithms as I did, with some variations.

3 Dataset and Features

I collected data directly from Expedia, a major travel website. I looked at five different destinations: Boston (BOS), Chicago (CHI), Portland (PDX), Los Angeles (LAX), and New York (LGA). For each of these five destinations, I considered the source airport San Francisco (SFO), and hourly requests were made to get all available flights in the three week range starting from that date. The date range of data collected is from 11/8 to 12/10.

I chose to use the current day of the week, the day of the week of the flight, the current time of day (categorized into four 6-hour time periods), the flight?s time of day, the number of hours until the flight, number of stops, the duration in minutes, and the price of the flight as features. In some of the experiments, I used one hot labels to encode the day of week and time of day features as 7 and 4 different binary features, respectively.

To assign labels, I looked for whether the minimum price for a flight with the selected date and destination would decrease. While I collected data on all flights in each request in order to allow for future exploration of segmentation of experiments, for this project I chose to consider only the minimum priced flight for each request for a date and destination. This led to a total of 75000 examples. I split this data randomly with a 70-30 split for training versus test examples.

4 Methods

4.1 Logistic Regression

Logistic Regression was used as a simple regression model as a baseline on the data. With the sigmoid function as the hypothesis function

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

returning values in $(0, 1)$, logistic regression fits using maximum likelihood

$$L(\theta) = \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

to find the best fit weight vector θ.

4.2 SVM

SVM is a supervised learning model. The algorithm for SVM solves the following optimization problem:

$$\max_{\gamma,w,b} \gamma$$

$$s.t. y^{(i)}(w^T x^{(i)} + b) \geq \gamma, i = 1, \ldots, m$$

$$||w|| = 1$$

The kernel I used was the Radial Basis Function (RBF), which is as follows:

$$K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)}) = \exp(-\gamma ||x^{(i)} - x^{(j)}||^2)$$

4.3 K-Nearest Neighbors

K-Nearest Neighbors assigns each point to the most common category of its k nearest neighbors within the training set, by Euclidean (L2) distance. I selected the value of $k = 3$, generally this selection is based on maximization of accuracy while preventing sensitivity to noise and overfitting.

Table 1: Models and Training/Test Accuracies

Part		
Model	Training Set Accuracy	Test Set Accuracy
Logistic Regression	69.1%	69.0%
Logistic Regression (with one hot labels)	69.9%	69.4%
SVM	98.2%	94.1%
kNN (k=3)	95.9%	91.3%
Random Forest (n=18)	99.3%	95.2%

Table 2: Models and Test Pos/Neg Accuracies

Part		
Model	Test Accuracy for Positives	Test Accuracy for Negatives
Logistic Regression	50.7%	80.6%
Logistic Regression (with one hot labels)	51.8%	80.7%
SVM	81.95%	96.1%
kNN (k=3)	90.2%	92.0%
Random Forest (n=18)	93.1%	95.6%

4.4 Random Forest

In random forests, trees are built from random samples of the training set. For each tree, the best split is chosen when splitting a node. This is summarized in the following pseudocode [6]:

```
1 Randomly select k features from total m features, where k << m
2 Among the k features, calculate the node d using the best split point
3 Split the node into child nodes using the best split
4 Repeat steps 1-3 until l number of nodes has been reached
5 Repeat steps 1-4 n times to create n trees
```

I used $n = 18$, which maximized accuracy for both training and test set.

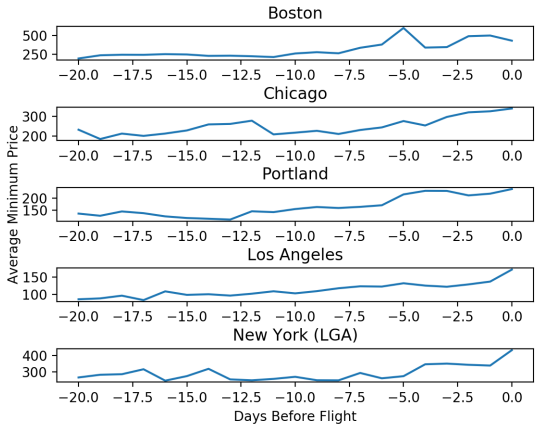
5 Results and Discussion

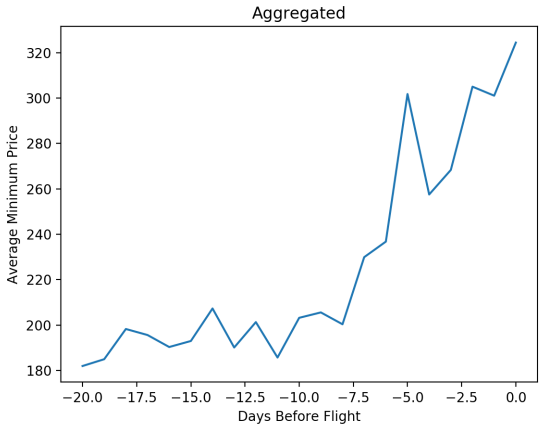
The initial model I tried as a baseline was logistic regression, which quickly showed that the data was likely not possible to model linearly, since the percentage of positives in the data was about 39%, so logistic regression had only slightly better performance than naively labeling all data points as negative (61% accuracy). Using the one-hot labels in this case also showed little benefit.

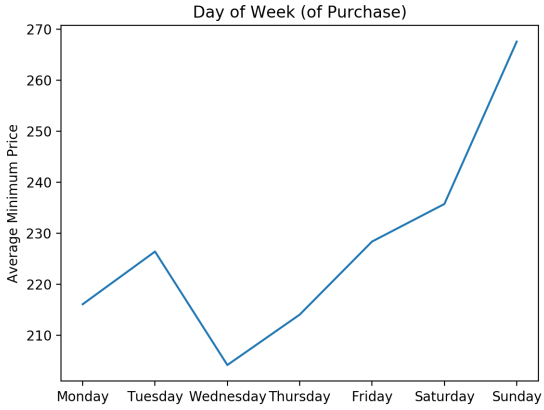
The classification models I tried (SVM, k-Nearest-Neighbors, Random Forest) performed well. However, the fact that all of the test set accuracies are a few percentage points less than their respective training set accuracies show that these models are overfitting the data. I also believe overfitting may have inflated the accuracies, since the test set data and training set data essentially have overlapping points since data was randomly split but flight pricing does not change hourly and the data points I collected are hourly; it is likely that data points taken from adjacent hours would contain much of the same information.

Another problem with these models is that they all perform better on negative cases than positive cases. In models for which precision values (accuracies for positives) are much lower than accuracy, like SVM and Logistic Regression, this means there are a lot of false positives. This indicates that the high accuracy values are less meaningful, since there are more negative examples than positive examples (39.06% of the training set and 38.91% of the test set are positives, which is the "should buy" case). The reason there are more negative examples is intuitive, it's likely that price generally increases as time goes on, so there should be more cases where the price does not drop than cases where the price does drop. However, Random Forest and kNN both do well in both accuracy and precision. Random Forest appears to be the best model overall.

Looking at some of the most simple features, we can also see some interesting trends in the data even without the models. The following graphs display the relationship between the price and the number of days in advance of the flight and the relationship between the price and the day of the week of the purchase. This shows that there is an obvious trend that prices on Wednesdays are significantly less than on Sundays.







6 Conclusion and Future Work

The next steps are to segment the data further in order to reach more interesting results. For example, I may need to depart after a certain time, or prefer to take a flight with no more than one layover. With more data collection, this would set this project apart from existing work in the field which considers either daily minimums as I did or specific flight number. I also want to continue to look at regularization and feature selection to avoid overfitting and get more robust results, and future work would require data collected over a longer time period with additional features to predict special circumstances such as holidays when flight pricing varies from usual. I would also like to do further feature selection to try to reduce computation time with models such as SVM while maintaining accuracy.

7 References

[1] Etzioni, O., Tuchinda, R., Knoblock, C. A., & Yates, A. (2003). To buy or not to buy: mining airfare data to minimize ticket purchase price. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 119-128). ACM.

[2] Papadakis, M. Predicting Airfare Prices.

[3] Tziridis, K., Kalampokas, T., Papakostas, G. A., & Diamantaras, K. I. (2017). Airfare prices prediction using machine learning techniques. In *Signal Processing Conference (EUSIPCO), 2017 25th European* (pp. 1036-1039). IEEE.

[4] Lu, J. (2017). Machine learning modeling for time series problem: Predicting flight ticket prices. *arXiv preprint arXiv:1705.07205.*

[5] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830.

[6] Polamuri, S. (2017). *How the Random Forest Algorithm Works in Machine Learning*. Retrieved from http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/