Machine learning modeling for time series problem: Predicting flight ticket prices

Jun Lu,
Computer Science, EPFL
jun.lu.locky@gmail.com,

Abstract

Machine learning has been used in all kinds of fields. In this article, we introduce how machine learning can be applied into time series problem. Especially, we use the airline ticket prediction problem as our specific problem.

Airline companies use many different variables to determine the flight ticket prices: indicator whether the travel is during the holidays, the number of free seats in the plane etc. Some of the variables are observed, but some of them are hidden.

Based on the data over a 103 day period, we trained our models, getting the best model - which is AdaBoost-Decision Tree Classification. This algorithm has best performance over the observed 8 routes which has 61.35% better performance than the random purchase strategy, and relatively small variance over these routes.

And we also considered the situation that we cannot get too much historical datas for some routes (for example the route is new and does not have historical data) or we do not want to train historical data to predict to buy or wait quickly, in which problem, we used HMM Sequence Classification based AdaBoost-Decision Tree Classification to perform our prediction on 12 new routes. Finally, we got 31.71% better performance than the random purchase strategy.[1]

1 Introduction

For purchasing an airplane ticket, the traditional purchase strategy is to buy a ticket far in advance of the flight's departure date to avoid the risk that the price may increase rapidly before the departure date. However, this is usually not always true, airplane companies can decrease the prices if they want to increase the sales. Airline companies use many different variables to determine the flight ticket prices: indicator whether the travel is during the holidays, the number of free seats in the plane etc., or even in which month it is. Some of

---

[1] A python implementation of this project is available online: https://github.com/junlulocky/AirTicketPredicting

the variables are observed, but some of them are hidden. In this context, buyers are trying to find the right day to buy the ticket, and on the contrary, the airplane companies are trying to keep the overall revenue as high as possible. The goal of this article is to use machine learning techniques to model the behavior of flight ticket prices over the time.

Airline companies have the freedom to change the flight ticket prices at any moment. Travellers can save money if they choose to buy a ticket when its price is the lowest. The problem is how to determine when is the best time to buy flight ticket for the desired destination and period. In other word, when given the historical price and the current price of a flight for a specific departure date, our algorithms need to determine whether it is suitable to buy or wait. In order to build and evaluate the model, we use data that contain historical flight ticket prices for particular routes.

2 Related work and our novelty

Some work has been done for determining optimal purchase timing for airline tickets. Our work is especially inspired by [Etzioni *et al.*, 2003]. Described in the paper, it achieves 61.8% of optimal. This result is very close to our result. However, our project goes beyond their work in several ways: the observation period is over a 103 day period (instead of a 41 day period); we extracted 8 routes for the prediction (rather than 2 routes in the existing work). This is a more difficult problem because over a longer period, the airplane companies tend to vary the price algorithm behind the company and they may have different price strategies for different routes.

Moreover, our novelty is that we extended the problem into regression and classification problems by some **model constructions**. Finally, given the historical datas and current data of the ticket, our system can predict to buy or to wait.

Another novelty is that, we also considered such a situation that some routes do not have any historical data, in which case we cannot perform the learning algorithms at all. This situation is very common, because there are always some new routes to be added by the airplane company or the company may conceal the historical data for some reasons. And also, this model can reduce computation time when we want to predict to buy or wait quickly because we do not need to train a large amount of data again. We call this problem as **generalized problem**. And on the contrary, for the previous problem, we call it as **specific problem**.
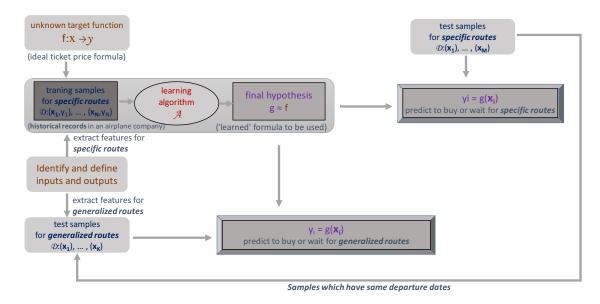
Figure 1: Work flow for the specific problem and generalized problem.

In the generalized problem, we were not given any historical data for these routes. But we were given the formula learned from specific problem and corresponding data which has same departure data with specific problems (which will be described more clearly in the section of Data Description and Interpretation). So we need to extract input features based on the test samples of generalized routes and test samples of specific routes as shown in Fig. 1. This model may have many benefits, especially when the historical data are not given or when we want to save time to quickly determine whether we should buy or wait for the new routes and not to spend too much time on model building for the new routes, and some other benefits such as decreasing data storage, interpretation and so on.

3 Data Collection

The data for our analysis was collected as daily price quotes from a major airplane search web site between Nov. 9, 2015 and Feb. 20, 2016 (103 observation days). A web crawler was used to query for each route and departure date pair, and the crawling was done every day at 10:00 AM.

For the purpose of our pilot study, we restricted the collecting data on non-stop, single-trip flights for 8 routes: 1 for Barcelona, Spain (BCN) to Budapest, Hungary (BUD); 2 for Budapest, Hungary to Barcelona, Spain; 3 for Brussels, Belgium (CRL) to Bucharest, Romania (OTP); 4 for Mulhouse, France (MLH) to Skopje, Macedonia (SKP); 5 for Malmo, Sweden (MMX) to Skopje, Macedonia; 6 for Bucharest, Romania to Brussels, Belgium; 7 for Skopje, Macedonia to Mulhouse, France; 8 for Skopje, Macedonia to Malmo, Sweden. And they are termed as R1 to R8. Overall, we collected 36, 575 observations (i.e. the queried price of each day for different departure dates and for the 8 different routes). In our observed airplane website, we did not find any tickets that were sold out in the specific queried days.

For the generalized problem, we also collected another 12 routes (i.e. BGY→OTP, BUD→VKO, CRL→OTP, CRL→WAW, LTN→OTP, LTN→PRG, OTP→BGY, OTP→CRL, OTP→LTN, OTP→LTN, PRG→LTN, VKO→BUD, WAW→CRL), which contains 14, 160 observations and termed as R9 to R20. And you can notice that two routes have already been observed in the specific problem, which are CRL→OTP and OTP→CRL. We keep these two routes to see how the generalized model can influence the performance. We will compare the results of the specific problem and generalized problem for these two routes especially.

3.1 Pricing Behavior in the Collected Data

We found that the ticket price for flights can vary significantly over time. Table 1 shows the minimum price, maximum price, and the maximum different in prices that can occur for flights of the 8 specific routes. In this table, although it's the maximum price and minimum price for all the departure dates for each route, we can have an overall glance at how we can achieve to decrease the ticket purchasing price. And Fig. 2a and 2b show how pricing strategies differ from flights.

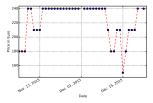## 4 Machine Learning Approach

4.1 Feature Extraction

The features extracted for training and testing are aggregated variables computed from the list of quotes observed on individual query days. For each query day, there are possibly eight airlines quoting flights for a specific origin-destination and departure date combination. For each query data, 5 features are computed, the *flight number* (encoded by dummy variables), the *minimum price so far*, the *maximum price so far*, the *query-to-departure* (number of

Route	Min Price	Max Price	Max Price Change
R1	29.99 €	279.99 €	250.0 €
R2	28.768 €	335.968 €	307.2 €
R3	9.99 €	239.99 €	230.0 €
R4	14.99 €	259.99 €	245.0 €
R5	15.48 €	265.08 €	249.6 €
R6	9.75 €	269.75 €	260.0 €
R7	16.182 €	332.982 €	316.8 €
R8	16.182 €	332.982 €	316.8 €

Table 1: **Minimum price, maximum price, and maximum change in ticket price for 8 specific routes. The decimals of the prices are from the currency change.**

(a) Price change over time for flight R4, departing on Jan. 13 2016. This figure shows an example of price drops to the minimum a slightly before departure date and have more price fluctuation.

(b) Price change over time for flight R3, departing on Dec. 22 2015. This figure shows an example of price drops to the minimum a slightly before departure date which may benefit the consumers and the high prices dominate.

Figure 2: The price flow of two representative route.

days between the first query date (09.11.2015 in our case) and departure date), the *days-to-departure* (number of days between the query and departure date), and *current price*.

For the output for regression problem, we set it to be the minimum price for each departure date and each flight; and as for the output for classification problem, we set the data entry of which the price is the minimum price from the query date to the departure date to 1 (namely Class 1 - to buy), otherwise, we set it to be 0 (namely Class 2 - to wait).

4.2 Data Description and Interpretation

Our data set consists of one set of input feature vectors, each one of them will be split into training dataset and testing dataset. In our case, we split the data that corresponds to flights with departure date in the interval between Nov. 9, 2015 and Jan. 15, 2016 as the training dataset; and the data that corresponds to flights with departure date in the interval betweeen Jan. 16, 2016 and Feb. 20, 2016 as the testing dataset; as for the generalized problem, we got the same time period as the test dataset, which is from Jan. 16, 2016 and Feb. 20, 2016.

Finally, the training dataset consists of N_{tr}=16, 208 data samples of one output variable y and input variable X. The testing dataset consists of N_{te}=20, 367, for which the output is unknown, and where we forged our predictions. The generalized problem testing dataset consists of N_{ge}=14, 160, for which the output is unknown as well, and we need to predict.

4.3 Model Construction

For regression, as the output is the minimum price for each departure date and each flight. Our regression method is to predict the expected minimum price for a given departure date and given flight with the input features. As a result, if the *current price* is less than the *expected minimum price*, we predict to buy; otherwise, we predict to wait. However, although it is very rare to happen, sometimes we may predict the *expected minimum price* of every entry to be smaller than the *current price*. Then, the last date should be seen as to buy. After we studied the data in depth, we were able to see that the last date always has a very high price. Thus we make the last buy date to be 7 days before departure date.

For classification, as our classification method is to predict to buy or to wait with the input features. As a result, if the prediction is 1, we buy the ticket, and we only buy the earliest.

In our approach, we used 7 machine learning models to compare the results, namely Least Squares for Regression [Harvey, 1966], Logistic Regression for Classification [Dayton, 1992], 3 layer Neural Networks, Decision Tree [Smith and Tansley, 2004], AdaBoost-Decision Tree [Freund and Mason, 1999], Random Forest [Liaw *et al.*, 2002], K Nearest Neighbors [Peterson, 2009], Uniform Blending [Lin, 2016] and Q Learning [Watkins and Dayan, 1992; Etzioni *et al.*, 2003]. For both the regression and classification problems, we tuned the hyperparameter via 5-fold cross validation (CV).

4.4 Performance Benchmarks

The naive purchase algorithm, called the *Random Purchase*, is to purchase a ticket randomly before the departure date. To be more concrete, for every departure date, for example departure date B, then from the first historical data for this departure date (namely date A), we pick several tickets randomly in this interval to simulate the clients buying ticket. And the average price would be computed as the *Random Purchase Price*. Another purchase strategy benchmark introduced in [Groves and Gini, 2013] is called *earliest purchase*, in which it splits the time interval into many day periods and it purchases one ticket in each day period. These two performance benchmarks are similar. However, we thought that the *Random Purchase* strategy conforms more closely to reality because in reality some clients may choose the same time period to buy rather than buy tickets in separated time interval.

The lowest achievable cost is called the *Optimal Price* and it is the lowest price between the first query date and the departure date.

5 Generalized Model

5.1 Uniform Blending

The simplest way to do the generalized predicting problem is using uniform blending. As long as we trained the specific problem, we could get 8 models (or learners) for 8 flight numbers separately. After that we can use the 8 models to predict for our new route, then we let these 8 models vote to buy or wait for every ticket.

5.2 HMM Sequence Classification

Apart from uniform blending, we can also allocate every data entry a "flight number" from the 8 routes, rather than average the 8 models. We then used sequence classification to allocate the "flight number" to every data entry.

In our problem, given a new observation sequence and a set of models, we want to explore which model explains the sequence best, or in other terms which model gives the highest likelihood to the data so that to extract corresponding features for the entry. And then, we can plugin the features from this chosen route to the new sequence to predict.

HMM sequence classification
Also called Maximum Likelihood classification. In practice, it is very often assumed that all the model priors are equal (i.e. that the new route to be predicted have equal probabilities of having same pattern in the observed 8 specific routes). Hence, this task consists mostly in performing the Maximum Likelihood classification of feature sequences. For that purpose, we must have of a set of HMMs [Eddy, 1996] that model the feature sequences. These models can be considered as **"stochastic templates"**. Then, we associate a new sequence to the most likely generative model. This part is called the *decoding* of the feature sequences.

Use HMM to solve our problem
We defined an **equivalence sequence** over different routes. Our equivalence sequence is the set of states with the same departure date, the same days before takeoff (i.e. current date or current entry to predict), and the same first observed date. Fig 3 shows how these 8 stochastic templates can be used to predict and intuitive meaning of **equivalence sequence**. Our goal is to allocate one pattern to each entry.
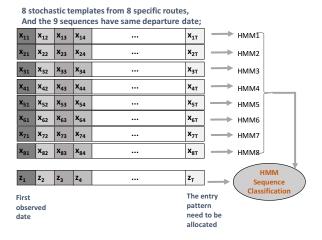

Figure 3: How to use 8 stochastic templates in HMM sequence classification.

In our case, as you know, we already have 8 patterns from the 8 specific routes. When we have to find out which pattern should be classified to the entries for a new route, we do a HMM Sequence Classification on the new sequence. In other word, we trained the 8 referenced sequence by HMM to get

each HMM model parameters Θ_i (i=1, 2, ..., 8), i.e. getting the transition matrix and emitting probability of each model. Then we compared it to the new sequence to get the *likelihood of the new sequence with respect to each HMM model*. Finally, allocate the pattern to the entry which makes it have the largest probability. This model is widely used in Automatic Speech Processing. To be more concrete, if an data entry is allocated to the second pattern of the 8 patterns. Then the dummy variable of flight number allocated to this entry should be: $F = [0,1,0,0,0,0,0,0]^\top$.

5.3 Performance Metric

As long as we get the *Randome Purchase Price*, *Optimal Price*, and the *Predicted Price*, we can use the following performance metric to evaluate our results:

$$Performance = \frac{Random\ Purchase\ Price - Predicted\ Price}{Random\ Purchase\ Price}\% \tag{1}$$

$$Optimal\ Perfor = \frac{Random\ Purchase\ Price - Optimal\ Price}{Random\ Purchase\ Price}\% \tag{2}$$

$$Normalized\ Performance = \frac{Performance}{Optimal\ Performance}\% \tag{3}$$

Having these metrics in mind, we used the Normalized Performance to evaluate our results, because it normalizes every route and it ranges from 0% to 100%, in which case, the higher the better, so that it gives more intuition about how well or bad is the result performance.

6 Experiments

6.1 Regression Results

Table 2 shows the results of regression methods. Random Forest Regression gets the best performance in regression method. However, it's variance is not small enough, which means it is sensitive to different routes. In this case, although for some routes, it gets good performance, for other routes, it gets bad performance. From the aspect of the clients, it is not fair for the some clients to buy tickets for which the system may predict badly. The preferred method in regression is AdaBoost-Decision Tree Regression method, which has smallest variance and a relative high performance.

6.2 Classification Results
Solving Imbalanced Data Set
Concerning a classification problem, an imbalanced data set leads to biased decisions towards the majority class and therefore an increase in the generalization error. As referred in the section of Data Description and Interpretation of the classification problem, the number of samples per class in our problem is not equally distributed, only few entries are to buy, most of the entries should be to wait. To address this problem, we can consider three approaches, namely **Random Under Sampling** (Randomly select a subset the majority classes' data points),

Performance(%)	Model								
Routes \ Method	Optimal	Random Purch.	Linear Regression	NN	Decision Tree	KNN	AdaBoost	Random Forest	Uniform Blending
R1	100.0	0.00	-42.17	67.03	42.31	38.19	48.49	54.67	44.37
R2	100.0	0.00	1.91	57.12	72.66	91.96	71.59	96.78	80.17
R3	100.0	0.00	10.34	18.60	30.40	37.48	41.01	56.35	25.68
R4	100.0	0.00	-21.02	-10.74	22.37	36.07	31.50	77.17	29.22
R5	100.0	0.00	-118.53	53.25	16.46	67.91	53.80	-0.69	65.70
R6	100.0	0.00	-19.47	57.30	63.85	69.09	64.11	67.51	63.85
R7	100.0	0.00	47.72	35.80	50.11	52.10	51.30	45.34	45.34
R8	100.0	0.00	-96.14	64.63	43.33	30.07	43.33	46.34	49.76
Mean Perf.	100.0	0.00	-29.67	42.87	42.68	52.86	50.64	**55.43**	50.51
Variance	0.00	0.00	2654.78	636.21	**37.04**	409.11	**143.49**	707.99	302.90

Table 2: **Regression Normalized Performance Comparison of 8 routes.**

Random Over Sampling (Randomly add redundancy to the data set by duplicating data points of the minority classes) and **Algorithmic Over Sampling** (Add redundancy to the data set by simulating the distribution of the data)

Having these methods in mind, firstly, the *Random Under Sampling* would not be useful in our problem, because in our problem, the data set is very unbalanced, i.e. the buy entries is very sparse. If we use *Random Under Sampling* [Japkowicz and Stephen, 2002; Japkowicz, 2000; Wasikowski and Chen, 2010; Guo *et al.*, 2008]. Secondly, if we use *Algorithmic Over Sampling*, it will add many noises into the data, because we do not know the hidden relationship between the features and the output. As a result, we preferred the second method, which is *Random Over Sampling*.

Identification of Outliers

We addressed the problem of outlier removal by making use of unsupervised learning methods [Chawla and Gionis, 2013] [Liu *et al.*, 2014], in particular through the implementation of K-Means and EM algorithm. Our approach was based on the fact that each characteristic class (either class 1 or 2) should in theory be restricted to a certain volume of the input space. Bearing this fact in mind, each of these two classes can be thought as a cluster. Consequently, we consider a sample to be an outlier if it does not belong to its labeled class cluster as illustrated by Fig. 4.

Having said that, we applied K-Means and EM to a training dataset. In the case of K-Means, 5896 samples were tagged as outliers. In the case of EM algorithm, 7318 samples were tagged as outliers. The choice of the initial conditions was crucial in terms of the algorithm's convergence and in terms of obtaining a reliable result.

Regarding the initial conditions, for our propose of use, we computed the initial center to be such that:
$\mu_{\mathbf{k}}^{\mathbf{initial}} = \frac{1}{\mathbf{N_k}} \sum_{\mathbf{x_n} \in \mathbf{C_k}} \mathbf{x_n}$, where N_k is the number of points belongs to Class k.

Fig 4 shows how the outlier can be found due to clustering methods. The cluster buy entries evolve in the cluster wait should be considered to be outlier, and vice-versa.

We tested our classification methods on both the datas with and without outlier removal. Then we chose the best result to show.
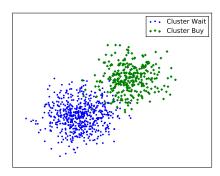

Figure 4: Outlier Removal.

Classification Performance Results

Table 3 shows the results of classification methods. As we see, AdaBoost-DecisionTree, KNN, and Uniform Blending get positive performance for all the 8 routes and have smaller variance over these routes compared to other classification algorithms. The AdaBoost-DecisionTree method gets the best performance and a relative low variance over 8 routes. And as expected, the uniform blending method has the lowest variance just like the theory of uniform blending describes.

6.3 Benchmark - Q Learning

We also implemented the method introduced in [Etzioni *et al.*, 2003] to compare. Table 4 shows the result of Q-Learning. As we see, the Q-Learning method described in [Etzioni *et al.*, 2003] has an acceptable performance and the variance is not large as well. The performance of it is very close to AdaBoost-DecisionTree Classification and Uniform blending Classification algorithms.

6.4 Generalized Problem Performance Result

Table 5 shows the result of generalized problem. As we can see, the uniform blending does not get any improvement. But the HMM Sequence Classification algorithm makes 9 routes get improvement, 3 routes have negative performance. Although the average performance is 31.71%, which is lower

Performance(%)	Model								
Routes \ Method	Optimal	Random Purch.	Logistic Regression	NN	Decision Tree	KNN	AdaBoost	Random Forest	Uniform Blending
R1	100.0	0.00	36.13	36.13	54.67	17.58	75.27	29.95	48.49
R2	100.0	0.00	39.97	39.98	11.02	64.62	60.34	86.60	30.32
R3	100.0	0.00	18.60	18.60	70.51	83.48	71.69	52.81	45.73
R4	100.0	0.00	-22.16	-22.15	57.76	38.35	32.64	0.67	46.34
R5	100.0	0.00	53.25	53.26	-26.42	34.44	50.76	-34.16	65.15
R6	100.0	0.00	57.30	57.30	80.35	87.69	85.33	19.83	71.71
R7	100.0	0.00	31.03	31.03	-28.2	55.28	65.02	33.61	49.12
R8	100.0	0.00	64.63	64.63	23.63	14.99	49.76	9.57	57.8
Mean Perf.	100.0	0.00	34.84	34.84	30.42	49.55	**61.35**	24.86	51.83
Variance	100.0	0.00	660.74	660.74	1592.45	679.48	151.25	1148.16	**144.56**

Table 3: **Classification Normalized Performance Comparison of 8 routes.**

Performance(%)	Model		
	Optimal	Random Purch.	Q Learning
R1	100.0	0.00	68.76
R2	100.0	0.00	61.81
R3	100.0	0.00	51.13
R4	100.0	0.00	54.01
R5	100.0	0.00	61.27
R6	100.0	0.00	72.08
R7	100.0	0.00	65.29
R8	100.0	0.00	6.50
Mean Perf.	100.0	0.00	55.11
Variance	0.00	0.00	380.06

Table 4: **Q Learning Performance.**

Perf. (%)	Model			
	Optimal	Random Purch.	Uniform	HMM
R9	100.0	0.00	80.52	87.09
R10	100.0	0.00	-75.1	-48.6
R11	100.0	0.00	51.06	63.11
R12	100.0	0.00	-16.57	17.32
R13	100.0	0.00	9.47	53.22
R14	100.0	0.00	21.14	45.83
R15	100.0	0.00	24.95	62.33
R16	100.0	0.00	-56.99	0.54
R17	100.0	0.00	-105.07	-84.45
R18	100.0	0.00	-0.18	38.16
R19	100.0	0.00	-48.61	-15.97
R20	100.0	0.00	-3.3	35.12
Mean Perf.	100.0	0.00	-14.84	31.71
Variance	0.00	0.00	2637.84	2313.25

Table 5: **Generalized Model Performance.**

than that of the specific problem, it makes sense that we did not use any historical data of these routes to predict (actually, there are two routes already appear in the specific problem, which are R3=R11 and R6=R16). In specific problem, when using the AdaBoost-DecisionTree Classification, the performances for these two routes (i.e. R3 and R6) are 71.69% and 85.33% respectively. However, in generalized problem, using same classification method, the performances are 63.11% and 0.54%, which has poorer performance than the specific problem. This is tolerable because we only used the formula trained in specific problem to predict.

7 Conclusion

In this article, we used the airplane ticket data over a 103 day period for 8 routes to perform out models. Removing outlier through K-Means Algorithm and EM Algorithm implied that our training algorithms were not influenced by non-representative class members; tackling the fact that our dataset was imbalanced, through Random Over Sampling, meant our algorithms were not biased towards the majority classes. For the classification and regression methods, the best values for hyperparameters were found through 5-fold grid search.

As shown by the results, for the **specific problem** and from the aspect of performance, AdaBoost-Decision Tree

Classification is suggested to be the best model, which has **61.35%** better performance over random purchase strategy and has relatively small performance variance for the 8 different routes. From the aspect of performance variance for different routes, Uniform Blending Classification is chosen as the best model with relatively high performance. On the other hand, the Q-Learning method got a relatively high performance as well. Compare the results of regression methods and classification methods, we could find that the cross validation error or precision in classification has far smaller variance than that of regression. We then considered that the classification model construction is more suitable in this problem.

For the **generalized problem** (i.e. predict without the historical data of routes that we want to predict), we did not test many models. However, the HMM Sequence Classification based AdaBoost-Decision Tree Classification model got a good performance over 12 new routes, which has **31.71%** better performance than the random purchase strategy.

For the generalized problem, we only used two methods to predict for generalized routes. In the future, we may find

more algorithms to see how can we extend the ticket prediction to generalized routes. Because this model may have many benefits, such as reducing computation time.

References

[Chawla and Gionis, 2013] Sanjay Chawla and Aristides Gionis. k-means–: A unified approach to clustering and outlier detection. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 189–197. SIAM, 2013.

[Dayton, 1992] C Mitchell Dayton. Logistic regression analysis. *Stat*, pages 474–574, 1992.

[Eddy, 1996] Sean R Eddy. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[Etzioni *et al.*, 2003] Oren Etzioni, Rattapoom Tuchinda, Craig A Knoblock, and Alexander Yates. 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*, pages 119–128. ACM, 2003.

[Freund and Mason, 1999] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133, 1999.

[Groves and Gini, 2013] William Groves and Maria L Gini. Optimal airline ticket purchasing using automated user-guided feature selection. In *IJCAI*, 2013.

[Guo *et al.*, 2008] Xinjian Guo, Yilong Yin, Cailing Dong, Gongping Yang, and Guangtong Zhou. On the class imbalance problem. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 4, pages 192–201. IEEE, 2008.

[Harvey, 1966] Walter R Harvey. Least-squares analysis of data with unequal subclass numbers. 1966.

[Japkowicz and Stephen, 2002] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[Japkowicz, 2000] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*, 2000.

[Liaw *et al.*, 2002] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[Lin, 2016] Hsuan-Tien Lin. Machine learning techniques. *Coursera*, 2016.

[Liu *et al.*, 2014] Wei Liu, Gang Hua, and John R Smith. Unsupervised one-class learning for automatic outlier removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3826–3833, 2014.

[Peterson, 2009] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[Smith and Tansley, 2004] Laurence Smith and John Tansley. Decision tree analysis, October 7 2004. US Patent App. 10/406,836.

[Wasikowski and Chen, 2010] Mike Wasikowski and Xuewen Chen. Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on knowledge and data engineering*, 22(10):1388–1400, 2010.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.