Flight Price Prediction and Feature Analysis using Machine Learning Techniques

Fink, Nicolas
nfink7@gatech.edu

Karimi, Araz
akarimi8@gatech.edu

Lin, Yifan
ylin498@gatech.edu

Liu, Yiying
yliu3312@gatech.edu

I. Project Summary

In this project, we have built several regression models on a data set that contains general info of 300,153 Indian flights.

The primary objective is to predict the price of the flights using different algorithms and discuss their difference, this is meaningful for learning purposes. The second objective is to interpret the linear model, which is helpful for answering real life questions like "How can I save on my flight?".

In the end, our models have achieved an accuracy of over 0.90 in R^2 measure, nonlinear models are especially more accurate. Our analysis has revealed the importance of different factors of price to a quantitative extent.

II. Tasks Implementations

The project consists of six major tasks and analysis, at least one member is responsible for each task. Each of the tasks leaders and project milestone responsibilities are listed below:

- Fink: Task 2, Task 3, Feature Analysis, Video Presentation, Final Report.
- Liu: Task 4, Video Presentation
- Lin: Task 1, Task 6, MLP, Video Presentation, Final Report
- Karimi: Task 5, Video Presentation

A. Task 1: Data Pre-Processing

Firstly, we converted the categorical values in the data set into real values using mainly one-hot encoding for the purpose of building regression models. Secondly, we split out a training set of 25,000 samples whose size is both sufficient and tractable for most of our candidate algorithms. The rest of the data went into the test set. Finally, as a common practice in machine learning problems, we standardize the data set using estimates on the training set.

B. Task 2: Principal Component Analysis

Learning how the features influence each other can be useful, but in the scope of machine learning, it can also be important to know how the features can be used to predict our target value. Principal component analysis (PCA) is usually used in data sets to reduce the amount of features from hundreds to something a little more manageable and computationally forgiving. While it is not essential to perform PCA on this data set, it can be good to know when time can be saved computing and how our features retain variance in the data set.

In our experiment we used the Huber Regressor (explained in more detail in "Robust Regression"), to test feature reduction. In PCA, by reducing the amount of components, you are removing some of the retained variance of the data set. Remove too many components and any algorithm will not be able to accurately predict the target variable.

Looking at Figure 1 (left), we can see that as the number of principal components kept decreased (percentage variance retained),

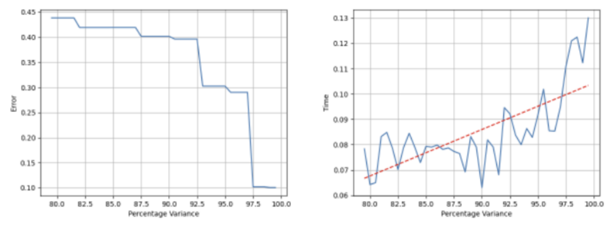

Fig. 1: Error (left) and comp. time (right) v. principal components

the error increased dramatically. The largest drop in error occurred when the regressor model was fit with 27 components (increased from 26 components). Looking at the right side of Figure 1, we can see that when more components are kept, computational cost increases. This tells us that if we had constraints on computational time, we could use 27 principal components of this data set and get a similar error in our regression analysis.

C. Task 3: Robust developed Regression

After the PCA analysis, the next step was to develop regression models: beginning with robust regression. The three standard regression models that we used were Least Squares Regression (LSE), Huber Regression (H), and Epsilon Insensitive Regression (ϵ). The loss functions for each are as follows:

$$L_{LSE}(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2 \qquad (1)$$

$$L_H(y_i, f(x_i)) = \begin{cases} \frac{1}{2}(r)^2, & \text{if } |r| < c \\ c|r| - \frac{1}{2}c^2, & \text{otherwise} \end{cases} \qquad (2)$$

where: $r = y_i - f(x_i)$

$$L_\epsilon(y_i, f(x_i)) = max(0, |y_i - f(x_i)| - \epsilon) \qquad (3)$$

Using these loss functions, we fit a model to the training data and minimized the training error on the (by tuning parameters) on the validation set before using it to predict the target variables from the test set. For our LSE regressor, we found a testing error of 0.0907 (parameter α = 0). For our Huber Regressor, we found a testing error of 0.0905 (parameters α = 0, c = 2.875). And finally, our ϵ-insensitive regressor was found to have a testing error of 0.0910 (parameters α = 78.47 and ϵ = 5310). Because we know that these regressors all treat outliers in our data set differently (Huber is less influenced by outliers, while LSE is more-so), it is interesting that they all perform with roughly the same testing error. This indicates that there may not be many outliers in our data set.

Because these regressors are linear, we were able to look into the coefficients of these models to determine how each feature was contributing to the flight price prediction. This analysis is covered in a later section (Section III-B3). After implementing these basic regressors, we were able to move onto more complex methods in an attempt to find a more accurate model.

D. Task 4: Regularized Regression and Kernel methods

In this task, we first implemented ridge regression and LASSO regression, which are linear regressors with a penalty term of L2 norm and L1 norm respectively. Then we tried out different kernel functions for the kernel ridge regression. Table I shows the performance of these regressors under our choice of hyperparameters.

Regressor	Regularizer Coef.	Param.	R^2 score
Ridge	3.7927	/	0.9099
LASSO	11.2884	/	0.9099
Poly kernel	5623.41	$deg = 3$	0.9620
RBF kernel	0.1	$\gamma = 0.0316$	0.9676

TABLE I: Performance of different regressors

From table I we can see that the ridge regressor and the LASSO regressor have quite similar performances which are nearly 0.91 in R^2 measure. This is not surprising given the fact that they are both linear regressors with low complexity.

Figure 2 visualizes the comparison between prediction from the LASSO regressor and true label on a subset of the test samples. We can see that the LASSO regressor gives better estimates for prices below 20,000 Rupees but predicts badly especially for prices over 60,000 Rupees. Similar pattern also shows up for the ridge regressor.

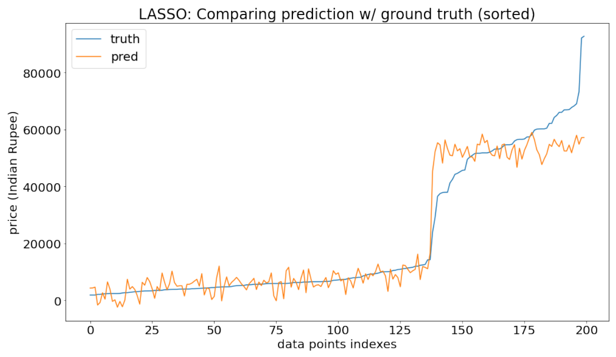

Fig. 2: Prediction from LASSO is not ideal especially for larger values

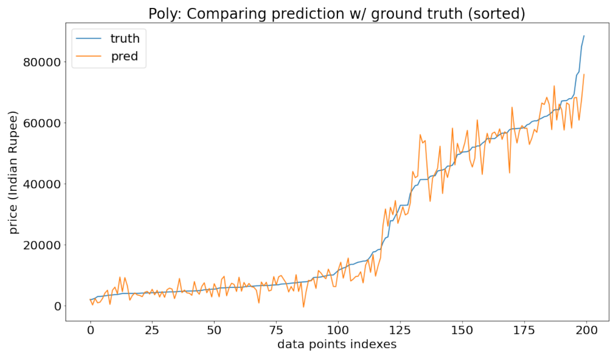

Fig. 3: Regressor with polynomial kernel of $deg = 3$

In comparison, kernel ridge regressors perform better on making predictions, for example as Figure 3 shows. However, it is noteworthy that the polynomial kernel can be numerically unstable as once discussed in the class, this can be illustrated by Figure 4.

In this example, the 4^{th} order polynomial achieved a R^2 score of 0.9821 on the training set but gets a horrible negative R^2 score on the validation set, meaning that the variance in its prediction is much larger than the variance of the original data, this can be seen by looking at those high peaks in the figure 4.

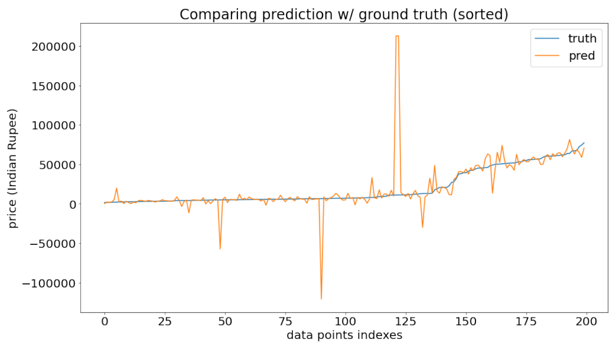

Fig. 4: Instability occurred when fitting with polynomial of $deg = 4$

E. Task 5: Support Vector Regression

In this task, we used a support vector regressor with different kernel functions to build our regression model. In comparison with the kernel ridge regressor, SVR uses only support vectors, which are "learned" from the training data, to predict target labels, making it more memory-efficient.

The core of this task boils down to tuning corresponding hyperparameters for different kernel functions: degree d for polynomial kernel, bandwidth γ for RBF kernel, and regularization coefficient C for all the kernels. Table II shows the performance of different kernel functions under our choice of hyperparameters.

Kernel	C	Param.	R^2 score
Polynomial	500	$deg = 7$	0.9620
RBF	10^5	$\gamma = 0.05$	0.9632

TABLE II: Performance with different kernels

Figure 5 visualizes the comparison between prediction from the SVM regressor and true label on a subset of the test samples. We can see that however SVM has an oscillatory results in all ranges, the error in all price ranges are nearly similar.

It is noteworthy though, the kernel method generally does not scale well to a larger data set. This is true for kernel ridge regression and also very true for SVR with nonlinear kernel function. In our case, the SVR function calling

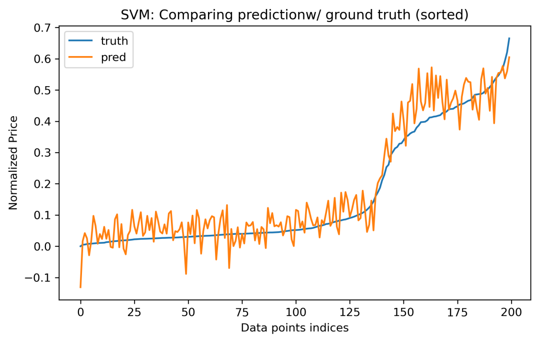

Fig. 5: Prediction from SVM vs true values of normalized prices

from the sklearn package took a long time to fit the training set, showing an obvious drawback.

F. Task 6: Random Forest

Random Forest is an ensemble learning algorithm of great potential for many machine learning problems. In this task, we built a Random Forest Regressor from the sklearn package. The function has several parameters to control its complexity but we mainly focused on three of them: the number of trees n, number of features used f, and maximum depth of the trees d.

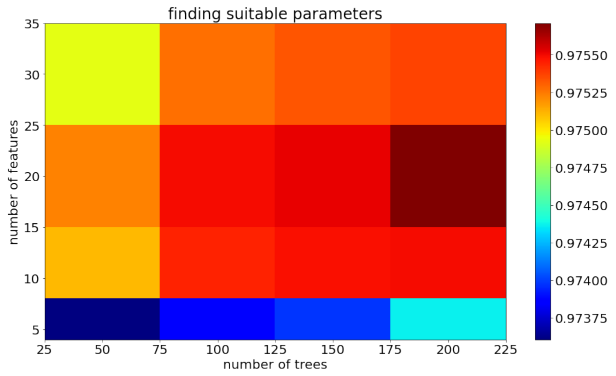

Fig. 6: Tuning of n and f at the same time

In principle, as n gets larger, the regressor will be more accurate but at a cost of longer training time; if f gets larger, the bias of the regressor decreases but the trees are more likely to be correlated; if d is too small, the model has a larger bias, but if d gets too large, the regressor is likely to over-fit and also takes longer to train. These three parameters are tuned through grid search and we used R^2 score to measure the performance.

Part of this tuning process can be illustrated by figure 6. This figure shows the R^2 score on part of the (n,f) parameter space from which we picked our "optimal" choice of the parameters. The maximum depth of trees d is fixed to 100. As we can see in the figure, having more trees will give higher accuracy (measured in R^2 score), whereas increasing the number of features to consider may lead to worse generalizability, as the trees are more likely to be correlated.

To sum up the grid search process, we found that the regressor can achieve a good accuracy of 0.9617 even with small values of (n,d,f) such as (10,10,20). From there, increasing n can improve the accuracy but the gain attenuates beyond certain point, and the run-time becomes prohibitive; increasing d leads to slight increase of run-time but no significant impact on the accuracy; whereas tuning f seems clearly a trade-off between lower bias and lower variance.

Eventually, we chose (n,d,f) to be (100,100,20) for a good accuracy, relatively low complexity, and reasonable run-time to train. The corresponding R^2 score in cross-validation and on the test set are 0.9757 and 0.9767, respectively.

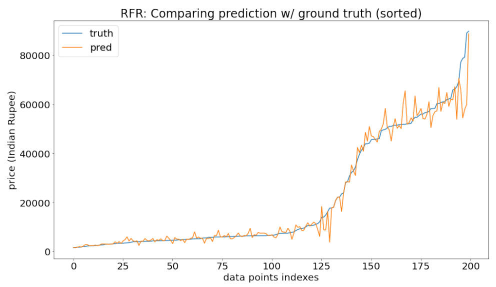

Fig. 7: Prediction is generally close to true label

Figure 7 visualizes the comparison between the prediction from the model and the true label on a subset of the test samples. From these results, we can see that although the prediction is more accurate for smaller values, in general it still fits the target price reasonably well. The random forest regressor also provided impurity-based feature importance, which will be discussed in section III-B2 as a part of the feature analysis.

III. Discussion

Table III summarizes the performances of all the regression models we've built so far. In this section, we will compare these algorithms and discuss their pros and cons. Then we will further analyze the importance of different features to the target label and try to interpret our linear model.

A. Algorithms

Algorithms	Score (R^2 test set)
Robust Regression (Huber)	0.9095
Ridge Regression	0.9099
LASSO	0.9099
KRR (poly)	0.9620
KRR (RBF)	0.9676
SVR (poly)	0.9628
SVR (RBF)	0.9632
Random Forest	0.9767
Multi-Layer Perceptron	0.9609

TABLE III: Performance of all models so far

We can see from the table that linear regressors all have a similar performance of close to 0.91 in R^2 score, while nonlinear regressors all have an accuracy of over 0.96. Comparing their scores, we can see that nonlinear models are generally more accurate than linear models, which is not surprising since nonlinear models are more complex and have larger VC dimensions. From that, we also have reasons to believe that different factors contribute to the price in a way that is nonlinear to some extent.

Other than the difference in accuracy, it is noticeable that linear regressors are much faster in terms of training process. In particular, when it comes to kernel methods, the cost of training can be significantly higher even to a prohibitive level. This is again not very surprising in view of the inherent difficulty to solve for a nonlinear regressor. However, random forest and multi-layer perceptron seem to have a more reasonable training time than other nonlinear regressors, making them a preferable choice for nonlinear model.

It is generally true that kernel methods do not scale well to larger data sets. Allegedly, the time and memory it costs to train is proportional to the number of samples squared. In our case, it sometimes took hours to train a model with kernel function. In contrast, the random forest regressor does not seem to suffer from larger data sets, not to mention those linear models.

Nonlinear models are generally hard to interpret since the nonlinear combination of different features seldom makes sense in the real world. Random forest arguably has limited interpretability for its intuitive structure of individual trees and the impurity-based feature importance that it provides. Meanwhile, linear models are easier to interpret although their coefficients may differ due to regularization.

B. Feature Analysis

1) Pearson Correlation: For the second part of our project, we wanted to tackle the more interesting and tangible questions brought up by this data set. How can we quantitatively address these features and their impact on the price of the flight? Using the Pearson standard correlation coefficient, we can calculate how each feature is correlated with every other feature.

Corr. Feature	Corr.
Class	0.937888
AL Visatra	0.358830
Duration	0.213861
Stops	0.123668
Days Left	-0.089003

TABLE IV: Price Correlations

Looking at Table IV, we can see that the highest correlated feature with price is class, followed by Airline Visatra, duration / class and then days left. This is a fairly intuitive conclusion, but to further analyze these features, we used our random forest impurities as well as our linear regression coefficients.

2) Random Forest Impurity Analysis: The random forest regressor tested as the most accurate model in predicting flight price but it also provided us with feature impurity measurements, which gave us more insight as to which features are more important than others. This was useful because the impurity measurements are a more nuanced way of confirming our correlation analysis while using our random forest regressor.

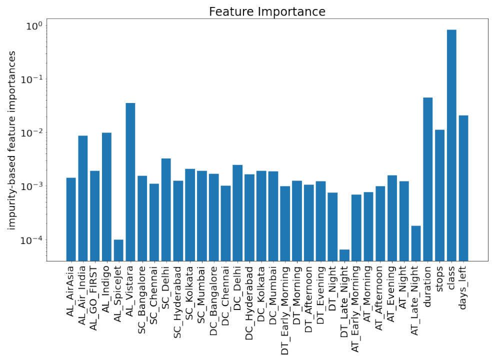

Fig. 8: Impurity Based Feature Importance

Figure 8 shows us that class, followed by duration, Airline Visatra, stops, and days left before flight are confirmed to be the most influential in flight price. Because random forest regressors aim to maximize the accuracy of the model through pure splits of the data, it is intuitive that our most accurate model would have a good idea of which features are more important in predicting our target variable.

3) Linear Model Interpretation: Through our correlation analysis and our random forest impurity analysis, we were able to gather which factors had the most effect on our final flight price. Using the average coefficients of our three robust regression techniques, we tried to quantitatively figure out the "cost" of each feature on the final flight price.
Table V shows us the amount of Rupees that would increase or decrease the price of the flight. For instance, getting a ticket from Airline Visatra would increase the average

Factor	Impact on Price
Average Price (Offset)	20624.4 (Rupees)
Class	+20646.2 (Business)
Duration	+584.6 (hours)
AL Visatra	+994.8
Days Left	-1736.0 (per days)
...	...

TABLE V: Regression Factors

price of that flight by approximately 994 Rupees, while purchasing a ticket an extra day before the scheduled departure would decrease the cost by 1,736 Rupees. With this analysis, we can quantitatively assign every feature with an estimate of how much someone would spend or save when purchasing a flight.

IV. Conclusion

A. Summary

In our research, we developed several algorithms to predict the flight price considering many factors. We found that the linear regressors were fast in computing and fairly accurate in prediction but were also useful in feature interpretation analysis. As we added more complexity to our regressors, the accuracy increased. Moving into the nonlinear space, we saw a significant jump in our scores, but lost the ability to interpret the model. We finally moved on to complex models such as a multi-layer perceptron network and random forests, where we found our random forest regressor to be the most accurate model for prediction.

We also tried to interpret the feature space by doing a principal component analysis on the data set, and concluded that 27 principal components could be used to run our regressors while maintaining a high accuracy in prediction. Next, we concluded through our correlation and random forest impurity analysis that we could identify the features that most impacted the price to a few key features: class, AL Visatra, duration, stops, and days left before flight. Using the coefficients of the linear analysis we were able to quantify these impacts on the flight price.

B. Future Work

In our analysis of this data set, we noticed that, overwhelmingly, the class feature has a significant impact on the data set. We believe this perhaps skewed the relative impact that other factors may or may not have had on the flight price. If we were to perform a similar analysis using only a subset of the data (i.e. data only from business class, or only from economy class) then we could get a better idea of which of the other features influence the price. More future work could also imply improvements to our existing algorithms to make them even more accurate at flight price prediction.