Analysis of the COVID effect on the airline passengers' demand

Data Science for Business Analytics

Team:
Ling-I Huang (lih238@nyu.edu)
Yu-Ting Chien (ytc354@nyu.edu)
Prudhvi Ghanta (prudhvi.ghanta@stern.nyu.edu)
Pradnya Wakchaure (pw1178@stern.nyu.edu)

Business Understanding

Business Problem: The airline industry has the lowest profit margins and a black swan event like the coronavirus pandemic can be catastrophic to the financial health of the company. The airlines will have to decide on which routes to focus the most on, to limit losses.

Use Scenario: Given a certain set of details like route, origin, destination, and the COVID rates in those regions, we can predict the demand (number of passengers). Airline companies can use this prediction to manage operations.

Data Mining Problem (Supervised): Predict the number of passengers between different routes using various linear regression techniques. A number of features like past travel data for the last 20 years, and the COVID cases in those respective places, will be used, and relevant features will be extracted from the dataset for a meaningful and useful prediction.

How exactly would this technical solution solve the problem? Airline demand is the number of passengers that use domestic, and international flights between different routes. Given the current COVID situation, if the covid infection rate is taken into account, and cost related analysis is added to the infection rate, optimal routes can be picked to maximize revenues, and thereby profits. The trained models can be expanded to any other non-COVID situation as well, as airlines demand, in general, depends on a variety of macroeconomic factors.

Data Understanding

Data Source: We used two data sources: 1) *T-100 Segment (All Carriers)* table of *The Air Carrier Statistics* database from Bureau of transportation statistics, 2) *Time series summary* tables from COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University.

The first source contains certificated U.S. air carriers' traffic information for both international routes and U.S. domestic routes. The data is collected from the air carriers' monthly reports.

The aviation industry and press frequently use this database to analyse carrier market share, passenger, freight, and mail cargo flow. The database covers certificated U.S. air carriers with annual operating revenues of $20 million or more. The data is monthly and is available from 1990 to 2020. The last updated data we used is until June 2020.

The second source contains COVID-19 statistics for countries all over the world. The *time series summary* tables include the number of confirmed, death, and recovered cases for each region, updated once a day.

Data Instance: Each data instance in our dataset represents a specific route (origin/destination combination) of a specific air carrier for a certain month. For example, one record could represent the aggregate information of all the flights of Delta Airline that is from New York to Seattle in January 2020.

Target Variable: Our target variable is the number of passengers for each record. That is, the total number of passengers for a certain route of a specific airline in a specific month.

Features: We included the number of passengers for previous years and the number of confirmed/death cases of the origin and destination to help us predict the target variable.

Nominal/String:

1. The origin/destination state/country (state is for U.S Domestic data, country is for international data):
DEST_STATE_ABR/DEST_COUNTRY, ORIGIN_STATE_ABR/ORIGIN_COUNTRY
2. Distance of the flights, grouped every 500 Miles: *DISTANCE_GROUP*
3. *MONTH,*
4. Identification of different air carriers: *UNIQUE_CARRIER*

Continuous:

1. The number of passengers of the same (route, carrier, month) combination for previous years:
PASSENGERS_00 (2000), PASSENGERS_01 (2001), PASSENGERS_02 (2002), PASSENGERS_03 (2003), PASSENGERS_04 (2004), PASSENGERS_05 (2005), PASSENGERS_06 (2006), PASSENGERS_07 (2007), PASSENGERS_08 (2008), PASSENGERS_09 (2009), PASSENGERS_10 (2010), PASSENGERS_11 (2011),

PASSENGERS_12 (2012), PASSENGERS_13 (2013), PASSENGERS_14 (2014), PASSENGERS_15 (2015), PASSENGERS_16 (2016), PASSENGERS_17 (2017), PASSENGERS_18 (2018), PASSENGERS_19 (2019),

2. The aggregate number of confirmed/death cases for the origin of the flights:

a. Total number of confirmed/death cases at the origin/dest for the month:
ORIGIN_monthlySum_confirmed, ORIGIN_monthlySum_death, DEST_monthlySum_confirmed, DEST_monthlySum_death

b. Total number of confirmed/death cases for each of the week of the month:
ORIGIN_1thWeekSum_confirmed, ORIGIN_2thWeekSum_confirmed, ORIGIN_3thWeekSum_confirmed, ORIGIN_4thWeekSum_confirmed, ORIGIN_1thWeekSum_death, ORIGIN_2thWeekSum_death, ORIGIN_3thWeekSum_death, ORIGIN_4thWeekSum_death, DEST_1thWeekSum_confirmed, DEST_2thWeekSum_confirmed, DEST_3thWeekSum_confirmed, DEST_4thWeekSum_confirmed, DEST_1thWeekSum_death, DEST_2thWeekSum_death, DEST_3thWeekSum_death, DEST_4thWeekSum_death

c. Daily average number of confirmed/death cases for the month:
ORIGIN_dailyAvg_confirmed, ORIGIN_dailyAvg_death, DEST_dailyAvg_confirmed, DEST_dailyAvg_death

d. Cumulative number of confirmed/death cases at the origin/dest since January 2020:
ORIGIN_cummulativeSum_confirmed, ORIGIN_cummulativeSum_death, DEST_cummulativeSum_confirmed, DEST_cummulativeSum_death

Data Preparation

There are four main steps to prepare our data: 1) preprocess the Airline datasets, 2) preprocess the COVID-19 datasets, 3) join the COVID-19 related features to the Airline table based on the origin/destination of the record, 4) transform the categorical features into numerical values.

Preprocess the Airline datasets

For preprocessing airline datasets, our final goal is to get two tables, containing information for U.S. domestic routes and international routes, respectively. We downloaded the tables from the Bureau of transportation statistics website and uploaded them to our github repository.

To include previous years' data of number of passengers of the same (origin, destination, carrier, month) combination, we added a key column for each record in the airline information dataframes. This key is the combination of these columns' values: 'UNIQUE_CARRIER', 'ORIGIN_AIRPORT_ID', 'DEST_AIRPORT_ID', and 'MONTH'. For example:

UNIQUE_CARRIER	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	key
3S	12478	11760	3S_12478_11760_1
3S	12892	11760	3S_12892_11760_1
3S	13930	11760	3S_13930_11760_1
3S	13252	11298	3S_13252_11298_1
3S	11298	11760	3S_11298_11760_1

While validating whether this key is unique and can be used to do one-on-one join, we found that in some previous years, each record actually represents a single flight, with only a few hundreds of passengers. So we also need to group the rows with the same key to get the aggregate information as we expected. We sum up the number of passengers for each group, and use the first value we encountered for other columns.

Now, we can merge the number of passengers from the previous years to 2020's airline dataframe. Now we have a table that each record represents the unique combination of (origin, destination, carrier, month), and the corresponding number of passengers from 2000 to 2020, along with other information that was included in the airline datasets, such as DISTANCE, DISTANCE_GROUP.

Finally, we split up this dataframe into two dataframes, one for U.S. domestic routes, and one for international routes. This aspect is already identified by the column DATA_SOURCE, so it's easily done.

Preprocess the COVID-19 datasets

For the COVID-19 datasets, we are interested in different aggregation that could be calculated from the daily values. The original dataset looks like the following. So, we used regular expression `regex = '^' + month + '/..?/..'` to filter out the columns for each month.

Country	**1/22/20**	**1/23/20**	**1/24/20**	**1/25/20**	**1/26/20**	**1/27/20**	**1/28/20**	**...**
Vietnam	0	2	2	2	2	2	2	...
Sri Lanka	0	0	0	0	0	1	1	...
Taiwan	1	1	3	3	4	5	8	...
Nepal	0	0	0	1	1	1	1	...
Brazil	0	0	0	0	0	0	0	...

Then, for each month, we calculate the sum, average, cumulative sum till that month, and the sum of every week (7 days) of that month. We ignored the data for 29th, 30th and 31st to keep the consistency of the meaning of these columns. Each of these weekly-sum columns are a sum of 7 days' numbers.

After the calculation, we group the records based on the same State/Country. This is because for some international data, the records are on State/Province level, and for the U.S data, the records are on County level.

Eventually, we joined the data for death cases and confirmed cases on region (either country or state), because they were in separate tables initially. So, we get two tables in the end, one is the COVID-19 aggregate information for every States in the U.S. and one is the same for every Countries among the world. They look like the following, with the column name formed of `{month}_{aggregate}_{tag}`.

Country	1_monthlySum_confirmed	1_dailyAvg_confirmed	1_cummulativeSum_confirmed	1_1thWeekSum_confirmed	...
United States	41.0	1.3225	41.0	0	...
Angola	0	0	0	0	...
Afghanistan	0	0	0	0	...
Yemen	0	0	0	0	...
Zimbabwe	0	0	0	0	...

Join the COVID-19 related features to the Airline table

For joining U.S. domestic airline dataframe with COVID-19 features, we loop through each record of the airline table, and add the columns of corresponding COVID-19 features for them based on the region of their origin and destination. Afterwards, the COVID-19 features, such as *ORIGIN_dailyAvg_confirmed, DEST_dailyAvg_confirmed, ORIGIN_dailyAvg_death, DEST_dailyAvg_death*, are attached to the airline records.

One thing worth mentioning is that even though most of the State names and Country names are consistent among the airline data and the COVID-19 data, there are a few exceptions. We renamed some country/states names whether in the COVID-19 dataframes or in the airline dataframes. For example, *'Virgin Islands' vs 'U.S. Virgin Islands'.*

Also, some regions shown in *COUNTRY_NAME* column in the airline data are considered as *Province_State* in the COVID-19 data instead of *Country_Region* as we expected. For example, in the airline data, we can see ‘Hong Kong’ as *COUNTRY_NAME*, but in the COVID-19 data, ‘Hong Kong’ shows in the *Province_State* column with ‘China’ as its *Country_Region.* So, we rename the *Country_Region* column at COVID dataframe for these regions: *'Saint Maarten', 'Bermuda', 'Aruba', 'Turks and Caicos Islands', 'Hong Kong', 'Cayman Islands', 'French Polynesia', 'Curacao', 'Guadeloupe',* to keep the mapping procedure simpler.

Finally, we drop the useless records that don't have COVID-19 related features. This includes some airline records for the regions without proper COVID-19 data, including 'U.S. Pacific Trust Territories and Possessions' and 'Cook Islands'. Also, the records before March 2020, the outbreak of COVID-19 in the U.S. are also excluded.

Transform the categorical features into numerical values

The categorical features we have includes *DEST_STATE_ABR, DISTANCE_GROUP, MONTH, ORIGIN_STATE_ABR, and UNIQUE_CARRIER*. The *DISTANCE_GROUP* is already in an ordinal manner as its essence, so we only have to encode the other ones.

To transform the categorical features without generating too many dummy variables, we used target encoding technique with `TargetEncoder` in the `category_encoders` library.

Modeling and Evaluation

We trained our models in three iterations, and in each iteration we applied a feedback process of cleaning and training, and using the evaluation metrics in the feedback loop. Most of our training time was spent in training the models for domestic data, as domestic data has a number of useful records for training, and we could use some of the learning from domestic models, and apply it for international dataset.

We trained our data with a bunch of linear regression models available in scikit learn package, with 80% of data for the training, and 20% for the test set. We used K Nearest Neighbor Regression technique as our first model, and trained the rest of the models using KNN as our baseline. We used Linear Regression, Lasso Regression, Ridge Regression, ElasticNet Regression, and Random Forest Regressor. The parameter choices to control for complexity, overfitting, and results are in each subsection below.

<u>Scoring Metrics:</u>

We used a combination of Root Mean Squared Error (RMSE), R-squared value, and Explained Variance Score (EVS) as our scoring metrics to assess the models. The

R-squared on testing data for the Random Forest Regressor model that we built was 0.821. This indicates that 82.1% of the variation in passengers can be explained by the features in the model.

K Nearest Neighbor Regression

The K-Nearest Neighbor Regression requires normalized numeric features to calculate the euclidean distance between points before training the model. This model is computationally intensive, so we looked for an optimal K value. Our point of reference for the KNN model, and the other models was Root Mean Squared Error (RMSE), R-squared value, and Explained Variance Score (EVS). After testing with K values from 1 to 40, the optimal value was at K=3. The error metrics for the model is as follows:

	KNN
k	3
R-squared	0.5211
RMSE	2310.3290
EVS	0.5241

We did not pick KNN as our final model, as we could see a lot of improvement, with our final cleaned data set, with the other linear regression models. Also, the time complexity of the model does not make it an ideal one for a larger dataset.

Linear Regression

With the linear regression model, we could get a little better performance than the KNN model. R-Squared, RMSE, and EVS scores showed a little improvement with the same cleaned dataset that we used for the model. The metrics from training the model are as follows:

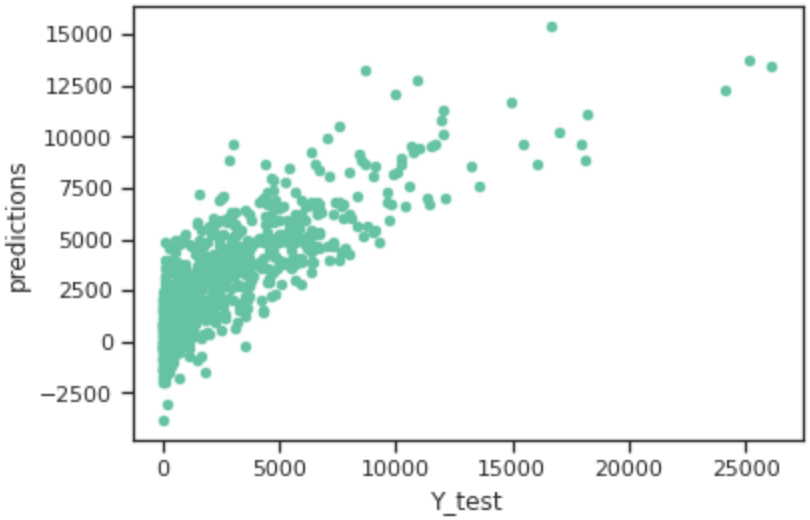
	Linear Regression
R-squared	0.6736
RMSE	1907.4096
EVS	0.6750

Lasso

The lasso regression is an extension of linear regression with L1 regularization. We tried with different parameters for alpha: 0.01, 0.1, and 1, and there was no significant difference between R-Squared, Root Mean Squared Error (RMSE), and Explained Variance Score (EVS). Also, we did not see any significant difference between the basic linear regression model, and the Lasso model.

	alpha=0.01	**alpha=0.1**	**alpha=1.0**
R-squared	0.6592	0.6592	0.6591
RMSE	1949.046	1949.052	1949.114
EVS	0.6605	0.6605	0.6605

The plot comparing the predictions and the expected values from the Lasso model is shown below:

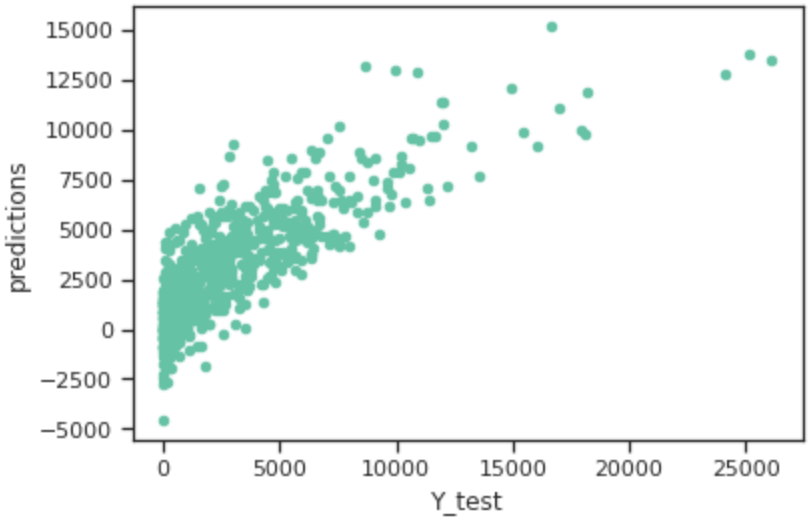


Ridge

The ridge regressor is also a linear model regression, but trained with L2 regularization (where coefficients are reduced to almost zero). We used RepeatedKFold, and GridSearchCV to check for different values for alpha, for the best alpha, between 0 and 1. The best value for our data set was 0.99. The trained model, when tested with our 20% test set, did give us better metrics than KNN, but not better than the linear regression model. Also, the performance with the ridge regressor was almost compatible with lasso regression. The metrics (with alpha=0.99) is as follows:

	alpha=0.99
R-squared	0.6736
RMSE	1907.4095
EVS	0.67502

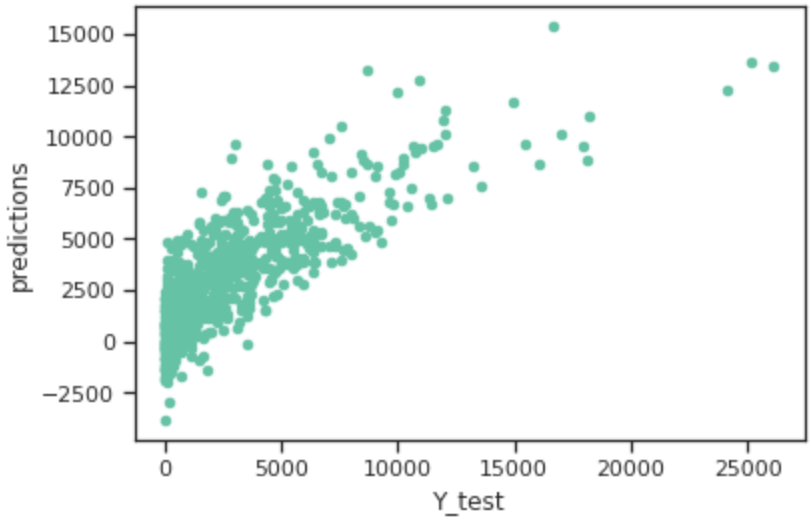
The plot comparing the predictions and the expected values from the Lasso model is shown below:



Elastic Net

The Elastic Net model is an extension of the linear regression model as well, and includes both L1 and L2 regularization. Both the penalty terms are affected by the alpha and L1 ratio parameters (a ratio of 0 indicates a pure Ridge model and a ratio of 1 indicates pure Lasso model). With alpha = 1.0, and l1_ratio = 0.5, the performance is not significantly better than the lasso or ridge models. In fact, it is almost compatible with both ridge and lasso models. The error metrics with the above mentioned alpha and l1 ratio, and the plot comparing the predictions and the expected values from the ElasticNet model is as follows:

	alpha=1.0, and l1_ratio=0.5
R-squared	0.65872
RMSE	1950.34070
EVS	0.658725



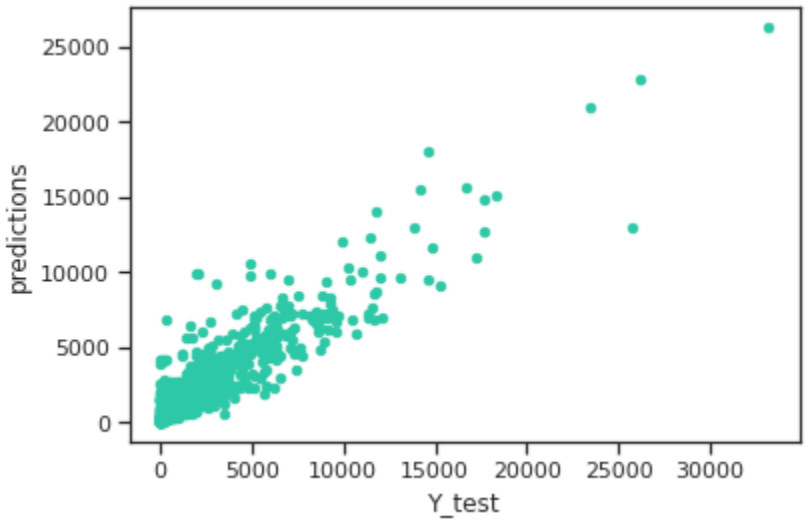
Random Forest Regressor

The Random Forest Regressor in brief is a combination of multiple decision trees trained on a slightly different (random) set of the observation data, with decision features that are also randomly selected for each tree. The final output of the forest is then the average of all the trees in an ensemble. The number of trees in the model is described by the number-of-estimators parameter. In layman terms, this regressor is comparable to our reliance on aggregate review scores for products rather than a single reviewer.

The training and a random forest regressor is thus computationally expensive and contains multiple parameters that can be tuned. To minimize computational time, we used both RandomizedSearchCV and GridSearchCV to identify trends in parameters for further optimization. If we optimize the model for the training data, then our model will score very well on the training set, but will not be able to generalize to new data, such as in a test set. When a model performs highly on the training set but poorly on the test set, this is known as overfitting, or essentially creating a model that knows the training set very well but cannot be applied to new problems. Therefore, the standard procedure for hyperparameter optimization accounts for overfitting through cross validation.

First, let's take a look at the result of the base model of Random Forest Regressor:

R-squared	0.81562
RMSE	1522.75865
EVS	0.8158760



For now, we only have a vague idea of the best hyperparameters and thus the best approach to narrow our search is to evaluate a wide range of values for each hyperparameter. The max_features parameter limits the number of features used in a decision node. The features that are selected at these nodes are random for every estimator tree generated. For these exploratory models, we applied “auto” in order to speed up computation time, assuming the underlying random selection still points in the right direction. The RandomizedSearchCV creates random combinations of the parameters to identify potential optimization areas, and the GridSearchCV exhaustively searches through every combination to find the largest score. The most important arguments in RandomizedSearchCV are n_iter, which controls the number of different combinations to try, and cv which is the number of folds to use for cross validation (we use 170 and 3 respectively). More iterations will cover a wider search space and more cv folds reduces the chances of overfitting, but raising each increases the run time.

The bolded parameter values are the optimal outputs of the search functions:

	RandomizedSearchCV	**GridSearchCV**
n_estimators	[2, **73**, 144, 215, 287, 358, 429, 501, 572, 643, 714, 786, 857, 928, 1000]	[70, 72, 74, **76**]
max_depth	[10, 20, **30**, 40, 50, 60, 70, 80, 90, 100, 110]	[35, 37, 39, 41, 43, **45**]
Training R-squared	0.8215	0.8137

Let's take a look at the result of the Random Search Cross Validation:

R-squared	0.8215
RMSE	1498.2076
EVS	0.8217

During the process, we achieved an unspectacular improvement on Training R-squared from 0.8156 → 0.8215. Depending on the application though, this could be a significant benefit. We can further improve our result by using grid search to focus on the most promising hyperparameters ranges found in the random search.

Let's take a look at the result of the Grid Search Cross Validation:

R-squared	0.8138
RMSE	1530.3521
EVS	0.8140

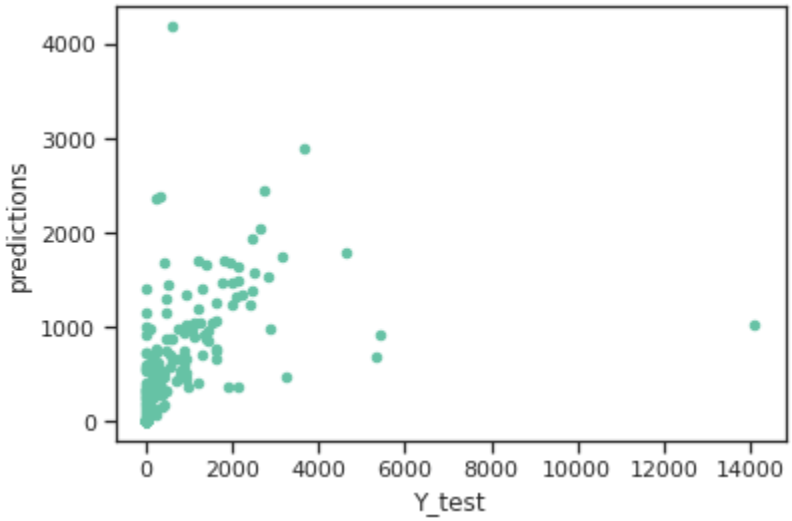
It seems we have maxed out performance in Random Search, a small decrease in performance of Grid Search indicates we have reached diminishing returns for hyperparameter tuning. We've already found the best parameters in random search and grid search shows that.

Modeling for international data

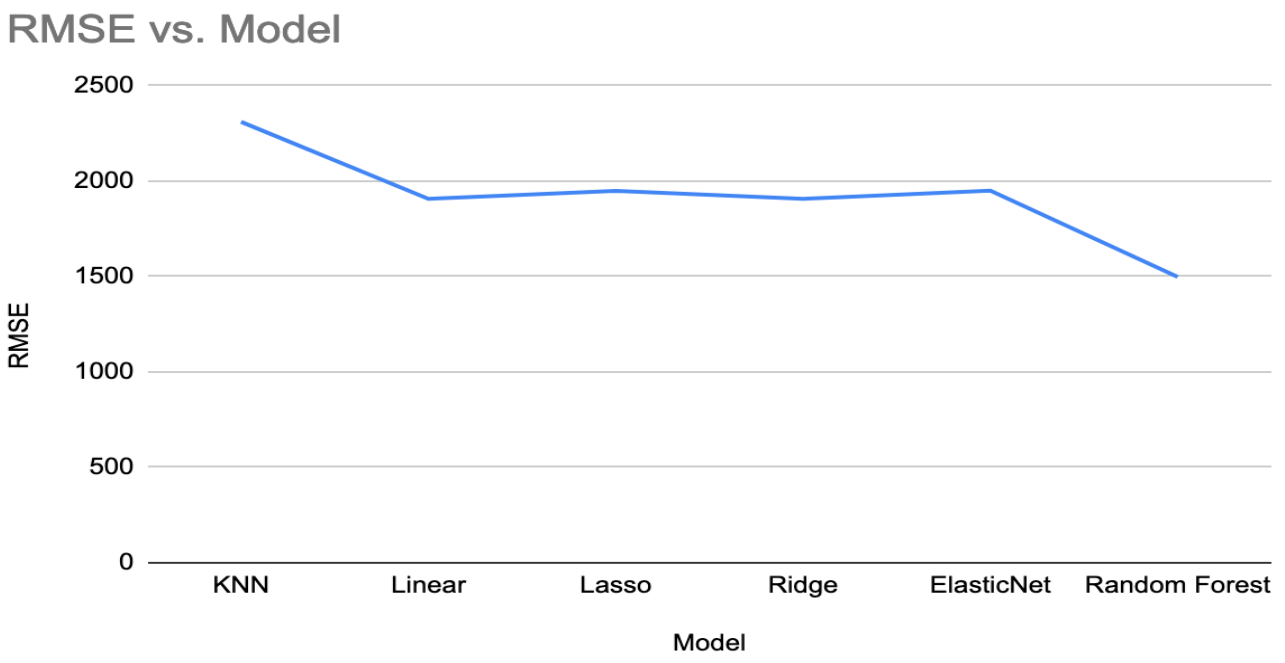
With Random Forest Regressor, we got the scores as follows. This is much lower than the performance of our model for U.S. domestic data. We believe this is because the number of instances of international data (972) is much less than the number with U.S. domestic data (3968).

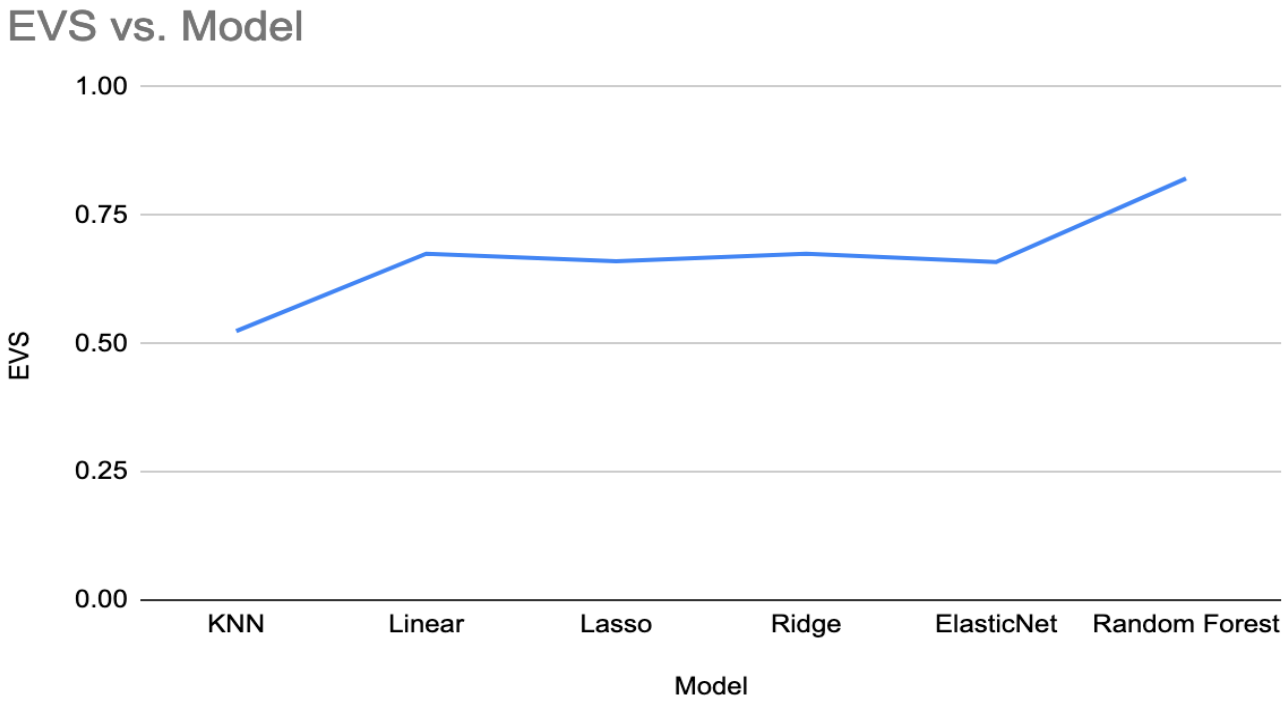
	alpha=0.01
R-squared	0.4689
RMSE	942.2408
EVS	0.4737

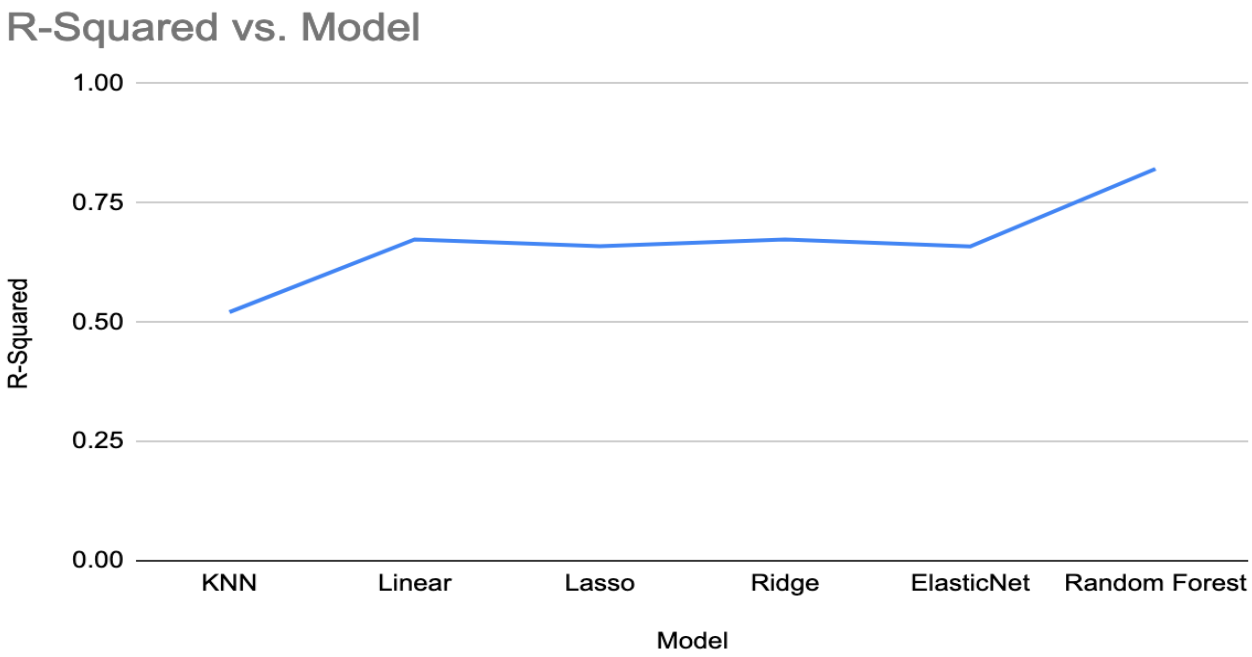
The plot comparing the predictions and the expected values from the model is shown below:



Model vs Metrics







Recommendations for Further Evaluation

Domain Knowledge Validation:

Domain experts who could help validate our model would be airline capacity planners and public health officials.

Projecting Expected Improvement:

Given we have seen a pandemic of this scale for the first time, the customer reaction for travel may not be identical if such a pandemic happens again. The best proxy that could be used here is the covid infection rate of a region which may not directly translate to the willingness to take flights. In reality there are many factors that will affect this, as mentioned in our issues to consider section below.

Deployment

Result Deployment:

To deploy our model we recommend a dashboard that will show the correlation between

the difference in passengers and the rate of covid infection in that region. This dashboard could also let a user select specific routes and carriers to have a more relevant passenger forecast. This will allow the users to see a general trend of demand for various regions.

We believe our models are useful to any stakeholders that would be interested to know airline passenger demand especially airline carriers. This model could also be enhanced for scenario planning for other catastrophic events that could affect demand. Airlines could apply their own metrics to calculate the future revenue based on their costs of operating specific routes. Since our model is based on public datasets it could also be used by travel booking companies to adjust their dynamic pricing algorithms. Other users of this model could be tourism industries, hotels, vacation property rentals.

Issues to Consider:

- Aircraft Capacity: Our dataset did not have information about the ratio of capacity to passengers. We expect this metric to have the most material effect in making business decisions.
- Seasonality: Our dataset has aggregate information for each route per month and per air carrier. Information about the seasonality within a month cannot be accurately forecasted using our models. For example, we have data for November but not for the week of thanksgiving when air travel is significantly higher. More granular weekly or daily data could make more nuanced predictions.
- Travel advisories of each region: The coronavirus pandemic saw different levels of government response across different regions. Several travel advisories were issued for many regions for different lengths of time that affect predictions.
- Business vs leisure travel: We don't have enough information to understand how business and leisure travel was affected and to what degrees.
- Willingness to fly: Passengers had a far lower willingness to travel during the earlier part of the epidemic compared to later. We observe residents of some states take more travel precautions than the others regardless of the infection rate.
- Rise in Cargo flights: In a pandemic while passenger demand would fall cargo

demand could improve we could enhance our dataset and models to include cargo and freight flight information.

Ethical Considerations:

Because of capacity forecasting if airlines have to dramatically lower their fleet, they could use it as an opportunity to price gouge its limited set of customers due to added financial pressures.

Risks:

Our model is mostly meant for illustrative purposes and based on public data. We have not captured the subtle differences in specific routes, carriers and aircrafts. This model is meant to provide a general picture and there is a chance that for some routes our predictions are way off.

We encourage the users of this model to understand that the model should be used for scenario planning purposes and to understand the general trends.

Measure Final Business Impact:

The primary use of the model would be in lowering operating costs because of low demand. Our model would predict a variable degree of lower demand of passengers both domestic and international. We expect this prediction will be used to estimate the lower operating capacity for an airline, airport or travel agencies. These stakeholders may want to operate on a limited capacity and reduce fleet size, personnel or certain low passenger volume flights to limit their losses. The final business impact would be the costs saved by lowering operational costs.

Appendix

Team Contributions

Ling-I Huang: Contributed to business topic and dataset selection, data understanding, data preparation sections with both code and written analysis

Yu-Ting Chien: Contributed to business topic and dataset selection, data preparation and mining, modeling code, hyperparameter tuning, and written analysis

Prudhvi Ghanta: Contributed to business topic and dataset selection, data understanding, data preparation, model evaluation,and written analysis

Pradnya Wakchaure: Contributed to business topic and dataset selection, built out business case with competitor research, coded visualizations, helped finalize written analysis

References

1. Encoding categorical variables
2. Evaluating a Linear Regression Model | Machine Learning, Deep Learning, and Computer Vision
3. Pandas Groupby: Summarising, Aggregating, and Grouping data in Python
4. Random Forest in Python
5. Sklearn Random Forest Regressor
6. Cross Validation
7. KNN
8. Accuracy metrics for regression
9. Improving the accuracy metrics