1. Background:

The purpose of this project is to take data on one-way flights taken in India in the year 2019 and use that to create a model that is able to predict the cost of a given flight. Using this information can inform us on what kinds of flights are most likely to give you the best deal.

2. Exploratory Data Analysis: EDA

2.1: Handling NaN values

Route, Total_Stops both have 1 missing value. These lie in the same row, so we can safely drop that row.

2.2: Analysis of Columns/Feature

Airline:

JetAirways by far is the most expensive airline. However, within each airline, there is a division between economy and higher classes like Premium Economy/ Business. Would be interesting to look into if that difference is more of a dividing factor than the airline itself. There are relatively few appearances of routes that do not take economy class.

Takeaways:

- Make a new feature indicating class. Is the impact of class significant to keep it despite not having that much data?

Date of Journey:

Unfortunately, the data seems to be limited in scope to the months of March-June, and that too with quite a small selection of dates (although no dates appear much less frequently than others). Our analysis may fall short because of this limited data.

Takeaways:

- Extract what day of the week or what month each flight took off and landed on. How does that impact the number of flights and the price of the flights?

Source/Destination:

We seem to be focusing on the largest cities in India. Delhi has the most flights taking off, and Cochin has the most flights landing.

Flights flying from Chennai by far are the cheapest flights, and Delhi by far the most expensive. However, Chennai only has non-stop flights, which explains why it's the cheapest. At each level of stops, Mumbai also has very expensive flights, but on the aggregate Delhi is more expensive.

In terms of destinations, Delhi and Cochin are the most expensive, and Kolkata is the cheapest. Again, New Delhi is the most expensive destination on the whole, but on each level of stops there are other cities that are more expensive. Is this just because some cities just have more flights that are not non-stop than others?

It appears that this is because some cities have more flights that are not non-stop. In this case, Delhi is more expensive because there are many more flights going to and from Delhi that make stops at other cities first. This makes sense, since New Delhi is India’s capital – there’s likely heavy traffic going to and from that city.

This dataset also doesn't contain any round trips (only one way tickets).
Takeaways:

- Could be interesting to look into whether there is a correlation between the population of the cities and the prices associated with the flights there.
- Can we find data regarding flights with Cochin as the source? It's likely that those numbers are quite high.

Route:
The routes are written in short form within a string. (Ex: 'CCU → IXR → BBI → BLR')

Takeaways:

- Should try and extract what the intermediate cities are. Price may depend on *which* cities a passenger is stopping over at.

Dep_Time:
Currently storing timing as strings.

Takeaways:

- Is there a way to better represent time (ie in categories rather than as numbers? Make a new feature indicating categories (overnight, early morning, morning, afternoon, evening, etc.)
- Do certain times have more flights than others and does that impact price?

Arrival Time:
Some values have a day written in the arrival, while some don't (Ex: *06:50 10 Mar* v. *13:15*). This indicates an overnight flight. Does that impact price?

Takeaways:

- Extract if a flight results in an overnight arrival.

Total Stops: the number of stopovers in this flight path
Currently storing this value as a string.

Takeaways:

- Convert to numbers from strings

Duration:
Duration of flight, Currently stored as strings in format of "__ hours, __ minutes".

Takeaways:

- Convert to integers (minutes).

Additional Info: Contains all other miscellaneous information, such as “No check-in baggage included”, “1 Short layover”, “1 Long layover”, or “Business class”.

Takeaways:

- Make each subsection of *AdditionalInfo* a different feature.
- Combine premium and business class into one feature since they are so similar, and such small features.

Price: this is the feature we are predicting. This is given in units of Indian rupees (₹).

2.3: Feature Engineering and Analysis of New Features

Date of Journey: created new features *DepMonth*, *DepWeekday*, *DepDay*

- April has significantly fewer flights, and significantly lower pricing.
- Mondays have the lowest prices.
- The 1st and 6th of the month have the highest prices, and prices tend to decrease as the month goes on.

Stopovers: created new features entitled *Stopover(CityName)* for each of the 42 cities that are ever a stopover. A 1 in that column indicates that there is a stopover in that city for that booking.

- Definitely a difference in prices based on the airport in which there’s a stopover. For instance, *StopoverSTV* bookings have an average price of 5746 rupees, while *StopoverGOI* bookings have an average price of 11708 rupees. A quick sanity check shows that this makes sense – GOI serves Goa, a popular tourist destination, and is thus much more likely to have more expensive bookings that go through it than a smaller airport like STV.
- Increasing from 0 to 1 stopovers doubles the price, but beyond that point there aren’t any significant jumps.

Number of Stopovers: converted values that were stored as strings into integers.

- Increasing from 0 to 1 stopovers doubles the average price, but beyond that point there aren’t any significant jumps.

Dep_Time: created features *HourDeparture*, *DeptMorn*, *DeptAfternoon*, *DeptNight*

- Doesn't seem to be a big difference between the categories (DeptMorn, DeptAfternoon, and DeptNight), although night flights (from 8pm to 4am) are the cheapest.

Arrival Time: created feature *HourArrival*, *ArrivMorn*, *ArrivAfternoon*, *ArrivNight*, *LandNextDay*

- Landing the next day has a significant impact on the price of the flights.
- There’s no significant difference between arriving in the morning, afternoon, or at night.

Duration: created feature *DurationMinutes*

- Increasing duration definitively increases the average price.

Additional Info: combined *Premium* and *BusinessClass*, created features *'AddInfo_1 Long layover'*, *'AddInfo_1 Short layover'*, *'AddInfo_2 Long layover'*, *'AddInfo_Change airports'*, *'AddInfo_In-flight meal not included'*, *'AddInfo_No check-in baggage included'*, *'AddInfo_Red-eye flight'*

Premium, in particular, has a significant impact on price.

- Would it be useful to make a column for long layovers, specifying 0/1/2 instead of having them be separated with the Add_Info columns? → No, there's only 1 flight each with 1 long layover or 2 long layovers

3. Training Models

3.1: Baselines + Hyperparameter Tuning

I selected 5 models for the regression problem: RandomForest, SVR, KNN, XGB, and MLP. Using the metric of mean absolute error (MAE), I ran a "baseline" cross validation with no specified hyperparameters for any of the models. Then, using random search, I tuned the hyperparameters for each of the models.

3.2: Results

The results for the models are found in column 2 of the table below:

Model	**Negative MAE w/ Cross-Validation (Section 1.4)**
Random Forest	₹ 626.08
SVR	₹ 2159.85
KNN	₹ 1658.52
XGB	₹ 661.23
MLP	₹ 1334.37

Table 1: Negative Mean Absolute Error (MAE) of Trained Models

Evidently, RandomForest and SVR are the most accurate models. In fact, the MAE of RandomForest is equivalent to about $7.50, which is quite good considering most ticket prices are at least 2 orders of magnitude greater than that.

A quick look through of the feature importances from RandomForest shows us that the analysis done during EDA is corroborated. For instance, duration is the most predictive feature. Other key features that were identified during EDA were *Premium*, *JetAirways*, and *landNextDay*.

Quite a bit of analysis was done on the sources and destinations of routes, and New Delhi was identified to be consistently expensive. Based on the random forest feature importance, if a route is in *any* way connected to New Delhi (starts, stops, or passes through the city), that information is also very predictive.

3.3: Feature Selection

Using the feature importance information from RandomForest, we can eliminate some features that contribute very little to the performance of the model. In this case, I selected the top 50% of features and retrained the optimal models from 1.4. We were able to get results that were only slightly worse than those of 1.4, with a marked improvement in the time it takes to train to get these results .

Model	**MAE w/ Cross-Validation**	**MAE w/ Cross-Validation + Feature Selection**	**Difference in MAE of columns 2 and 3**	**Training Time of Baseline**	**Training Time of Feature Selection Model**
Random Forest	₹ 626.08	₹ 633.18	₹ 7.10 ($ 0.10)	12.2s	10.1s
SVR	₹ 2159.85	₹ 2169.66	₹ 9.81 ($ 0.12)	60s	53.9s
KNN	₹ 1658.52	₹ 1686.12	₹ 27.6 ($ 0.34)	4.7s	2.3s
XGB	₹ 661.23	₹ 678.56	₹ 17.33 ($ 0.21)	15.9s	11.2s
MLP	₹ 1334.37	₹ 1474.18	₹ 139.81 ($ 1.70)	552s	270s

Table 2: Negative Mean Absolute Error (MAE) of Trained Models