Flight Price Prediction

Project Report

Author: Amer Tarek

January 2026

1. Executive Summary

This report presents an end-to-end machine learning project for predicting domestic flight prices in India. The project demonstrates the complete data science lifecycle, from initial data exploration and quality assessment through advanced modeling techniques to final deployment as a production-ready web application.

The dataset comprises 10,683 flight booking records from 11 major Indian airlines, covering routes between 5 source cities and 6 destination cities during the period of March to June 2019. Through systematic data cleaning and feature engineering, we transformed raw booking data into a high-quality dataset suitable for machine learning, retaining 97% of the original records while eliminating duplicates, missing values, and statistical outliers.

After evaluating different regression algorithms, XGBoost emerged as the best performer with a test R^2 of 0.9167 (91.7% accuracy) and a mean absolute error of just ₹659. This means the model can predict flight prices with an average deviation of less than ₹700 from actual prices, making it highly reliable for practical use.

Key Discovery: One of the most significant findings from this analysis is counterintuitive: non-stop flights are actually cheaper than connecting flights on average (₹4,999 vs ₹9,471). This challenges the common assumption that direct flights command a premium price, and the analysis reveals why—connecting flights have longer durations, which correlates positively with higher operating costs and thus higher prices.

The final deliverable includes a modular Python codebase with separate modules for data pipeline, model training, and inference, along with a Streamlit web application that allows users to get real-time flight price predictions by entering their travel details.

2. Phase 1: Data Understanding

2.1 Initial Data Exploration

The project began with a thorough examination of the raw dataset to understand its structure, quality, and characteristics. The dataset was provided as an Excel file containing flight booking information, with each row representing a single flight option available for booking.

Upon loading the data, we discovered a dataset with 10,683 rows and 11 columns. The columns included Airline (the carrier operating the flight), Date_of_Journey (travel date), Source (departure city), Destination (arrival city), Route (the path taken including any stops), Dep_Time (departure time), Arrival_Time (arrival time), Duration (total flight duration), Total_Stops (number of intermediate stops), Additional_Info (extra information like meal inclusion), and Price (the target variable we aim to predict).

2.2 Dataset Composition

The airline distribution revealed IndiGo as the dominant carrier with 24.5% of all flights, followed by Jet Airways at 15.9% and Air India at 15.8%. Multiple carrier options (codeshare flights) accounted for 14.7%, while SpiceJet represented 11.6% and Vistara 8.8%. The remaining carriers including Air Asia, GoAir, and Trujet together comprised about 8.7% of the dataset.

Geographic coverage showed Delhi as the primary source city, accounting for 42.6% of departures, followed by Kolkata (27.2%), Bangalore (15.9%), Mumbai (7.8%), and Chennai (6.5%). On the destination side, Cochin was the most common arrival city at 42.0%, followed by Bangalore (21.2%), Delhi (14.0%), New Delhi (11.7%), Hyderabad (6.5%), and Kolkata (4.6%).

The temporal distribution of flights showed March 2019 as the most represented month with 37.9% of records, followed by May (27.3%), June (19.9%), and April (14.9%). This four-month window provides a good representation of both regular travel periods and the beginning of the summer vacation season.

2.3 Target Variable Analysis

The target variable, Price, showed considerable variation in the raw data. Prices ranged from a minimum of ₹1,759 to a maximum of ₹79,512, with a mean of ₹9,084 and median of ₹7,662. The significant difference between mean and median (₹1,422) suggested a right-skewed distribution, which was confirmed by a skewness value of 1.81.

This high skewness indicated the presence of outliers on the upper end—extremely expensive flights that could potentially distort our model's predictions. The standard deviation of ₹4,611 further indicated substantial price variability across the dataset.

2.4 Data Quality Issues Identified

The initial exploration revealed several data quality issues that needed to be addressed before modeling. First, we identified 2 rows with missing values in the Route and Total_Stops columns. Second, we found 219 exact duplicate records (2.1% of the dataset), where all column values were identical. Third, the IQR analysis of the Price column identified 94 potential outliers (0.9%) with values above the upper bound of ₹23,024.

Additionally, the text-based columns required parsing and transformation. The Duration column contained strings like '2h 50m' that needed conversion to numerical minutes. The

Total_Stops column had values like 'non-stop', '1 stop', '2 stops' requiring mapping to integers. Date and time columns needed parsing to extract useful features like day of week, month, and hour of departure.

3. Phase 2: Data Cleaning & Feature Engineering

3.1 Cleaning Strategy

Based on the issues identified in Phase 1, we developed a systematic cleaning pipeline. The strategy prioritized data retention while ensuring data quality—we aimed to remove only genuinely problematic records while preserving as much valid data as possible.

The cleaning pipeline was implemented in a specific order to handle dependencies between steps. We first removed duplicates, then handled missing values, followed by outlier treatment. Feature engineering was performed next to create the derived columns needed for analysis and modeling.

3.2 Duplicate Removal

The first step removed 219 exact duplicate records from the dataset. These duplicates likely arose from data collection processes where the same flight option was recorded multiple times. Removing duplicates is essential to prevent the model from giving undue weight to repeated observations and to ensure the train-test split doesn't leak information. After this step, 10,464 records remained.

3.3 Missing Value Treatment

Only 2 rows contained missing values, specifically in the Route and Total_Stops columns. Given the extremely small proportion (0.02%) and the fact that these columns contain essential flight information that cannot be reasonably imputed, we chose to drop these rows. This decision had negligible impact on the dataset size while ensuring complete data integrity. The dataset now contained 10,462 records.

3.4 Price Outlier Removal

For the target variable Price, we applied the Interquartile Range (IQR) method to identify and remove outliers. The first quartile (Q1) was ₹4,218 and the third quartile (Q3) was ₹10,758, giving an IQR of ₹6,540. Using the standard 1.5×IQR rule, we calculated the upper bound as Q3 + 1.5×IQR = ₹23,024.

This process identified 94 flights with prices exceeding ₹23,024 as outliers. These extremely high-priced flights (some reaching ₹79,512) likely represented premium business class tickets or special circumstances not representative of typical flight pricing. Removing these outliers significantly improved the price distribution, reducing skewness from 1.81 to 0.45—a 75% improvement that created a much more normal distribution suitable for regression modeling. The dataset now contained 10,368 records.

3.5 Duration Outlier Detection

During the analysis phase, we discovered a critical data quality issue that required a novel solution. Some connecting flights had impossibly short durations—for example, a 1-stop flight with only 5 minutes total duration. This is physically impossible since even a single flight segment requires more time than this, not to mention the layover between connections. **This discovery led us to implement logical validation based on the number of stops.**

We established minimum duration thresholds for each stop category based on realistic flight operations: non-stop flights require at least 30 minutes (for very short domestic hops), 1-stop flights need minimum 2 hours (accounting for two flight segments plus a layover), 2-stop flights require at least 4 hours, 3-stop flights need 6 hours minimum, and 4-stop flights require at least 8 hours.

Applying this validation identified 5 invalid records: 3 one-stop flights and 2 two-stop flights with durations below their respective thresholds. These were clearly data entry errors and were removed. After a final duplicate check (to catch any records that became duplicates after outlier removal), the clean dataset contained 10,363 records—representing 97% of the original data.

3.6 Feature Engineering

Feature engineering transformed the raw columns into analysis-ready features. The Duration column, containing strings like '2h 50m', was parsed into Duration_Minutes (integer). For example, '2h 50m' became 170 minutes. The Total_Stops column was mapped from text to integers: 'non-stop' to 0, '1 stop' to 1, '2 stops' to 2, and so on.

The Date_of_Journey column was parsed to extract three new features: Journey_Day (1-31), Journey_Month (1-12), and Journey_Day_of_Week (0-6, where 0 is Monday). Similarly, Dep_Time was split into Dep_Hour (0-23) and Dep_Minute (0-59), and Arrival_Time was processed the same way.

We also created a Route_Pair feature by combining Source and Destination (e.g., 'Delhi_Cochin') for route-level analysis. The Additional_Info column was standardized to ensure consistent values. Finally, the original Route column was dropped as it contained detailed stopover information that would cause data leakage and was too granular for useful pattern extraction.

3.7 Cleaning Summary

Step	Records Removed	Remaining
Original Dataset	—	10,683
Duplicate Removal	219 (2.1%)	10,464
Missing Value Removal	2 (0.02%)	10,462
Price Outlier Removal (IQR)	94 (0.9%)	10,368
Duration Outlier Removal	5 (0.05%)	10,363
Final Clean Dataset	**320 total (3.0%)**	**10,363 (97%)**

The cleaning process successfully retained 97% of the original data while eliminating problematic records that could negatively impact model performance.

4. Phase 3: Exploratory Analysis & Business Questions

4.1 Univariate Analysis

Before diving into business questions, we conducted univariate analysis to understand the distribution of each variable independently. For numerical features, histograms revealed the distribution patterns. The Price variable, after outlier removal, showed a much more symmetric distribution with reduced right skew. Duration_Minutes exhibited a right-skewed distribution with most flights between 2-4 hours but a long tail extending to longer connecting flights.

The Stops variable showed that the vast majority of flights (71.7%) had exactly one stop, while non-stop flights comprised only 16.3% of the dataset. Two-stop flights accounted for 11.7%, and flights with three or more stops were rare at just 0.3%. This distribution reflects the reality of Indian domestic aviation where direct routes are limited and most journeys require connections.

For categorical features, bar charts visualized the frequency distributions. The airline distribution confirmed IndiGo's market dominance, while the source/destination analysis showed Delhi and Cochin as the primary traffic hubs in this dataset.

4.2 Business Questions Addressed

Phase 3 addressed eight key business questions through bivariate and multivariate analysis, combining statistical testing with visualization to derive actionable insights.

Question 1: How does the number of stops affect flight price?

This analysis revealed one of the project's most significant findings. Contrary to the common assumption that non-stop flights are more expensive due to convenience, our data showed the opposite pattern. Non-stop flights averaged ₹4,999, while one-stop flights averaged ₹9,471—an 89% premium for connecting flights. Two-stop flights were even more expensive at ₹10,525, and the rare 3+ stop flights averaged ₹18,792.

A t-test comparing non-stop versus connecting flights confirmed this difference is statistically significant ($p < 0.001$). The explanation lies in the relationship between stops and duration: more stops mean longer total travel time, and longer flights have higher operating costs (fuel, crew time, airport fees at multiple locations). Thus, the pattern is actually: more stops → longer duration → higher costs → higher prices.

Question 2: Which airlines are premium vs. budget carriers?

Analysis of average prices by airline revealed clear pricing tiers. Jet Airways emerged as the premium carrier with an average price of ₹13,097, followed by Vistara Premium Economy at ₹12,500 and Multiple Carriers Premium Economy at ₹11,200. These carriers commanded prices 40-60% above the market average.

The mid-tier included Air India (₹9,200), standard Vistara (₹8,900), and Multiple Carriers (₹8,500). The budget segment was led by SpiceJet at ₹4,479, followed by GoAir (₹5,100), IndiGo (₹5,800), and Air Asia (₹5,300). The price difference between the most expensive (Jet Airways) and cheapest (SpiceJet) carrier was 192%, demonstrating how significantly airline choice impacts ticket cost.

Question 3: What is the relationship between flight duration and price?

Scatter plot analysis of Duration_Minutes versus Price revealed a moderate positive correlation (r = 0.51). This means that as flight duration increases, prices tend to increase as well, though the relationship is not perfectly linear. The correlation explains about 26% of the variance in price.

However, when we controlled for the number of stops by analyzing the correlation within each stop category separately, an interesting pattern emerged. Within non-stop flights alone, the duration-price correlation was weaker, suggesting that for direct flights, duration (which largely reflects distance) is less important than other factors like airline brand and route demand.

Question 4: How do departure times influence price?

Grouping flights by departure hour revealed distinct pricing patterns throughout the day. Red-eye flights departing between midnight and 4 AM offered the lowest average prices, reflecting low demand for these inconvenient departure times. Morning flights (8 AM - 12 PM) commanded moderate prices, while evening flights (4 PM - 8 PM) were the most expensive, coinciding with business traveler preferences for end-of-day departures.

This pattern suggests that airlines implement time-based pricing strategies, charging premiums during high-demand periods and discounting flights at less desirable times to fill seats.

Question 5: Which routes are most and least expensive?

Route-level analysis using the Route_Pair feature identified significant price variations by origin-destination combination. The most expensive route was Bangalore to New Delhi, averaging ₹10,407. This high-traffic business corridor commands premium pricing due to strong corporate travel demand.

Conversely, the Chennai to Kolkata route offered the lowest average price at ₹4,790. The 117% price difference between the most and least expensive routes highlights how route selection can dramatically impact travel costs.

Questions 6-8: Temporal Patterns

Analysis of day-of-week patterns showed Friday as the cheapest day to fly (₹8,476 average) and Sunday as the most expensive (₹9,298). This approximately 10% weekend premium reflects leisure travel demand patterns where travelers prefer Sunday returns.

Monthly analysis revealed April as the cheapest month (₹8,102) and May as the most expensive (₹9,412). The May premium aligns with the beginning of summer vacation season when family travel increases demand.

Examining the interaction between time factors showed that the cheapest flights overall were Friday red-eye departures in April, while the most expensive were Sunday evening flights in May—demonstrating how multiple temporal factors compound to create significant price variations.

5. Phase 4: Machine Learning Preprocessing

5.1 Train-Test Split

With the clean dataset of 10,363 records ready, Phase 4 prepared the data for machine learning. The first step was splitting the data into training and test sets using an 80-20 ratio with a fixed random state (42) for reproducibility.

This resulted in 8,290 training samples and 2,073 test samples. We verified that the split maintained similar distributions by comparing mean prices: the training set averaged ₹8,842 while the test set averaged ₹8,787—a difference of only ₹55 (0.6%), confirming a well-balanced split.

5.2 Feature Selection and Leakage Prevention

Before building the preprocessing pipeline, we carefully considered which features to include. Three columns were identified as potentially causing data leakage or being redundant: Route_Pair (redundant since Source and Destination are already included separately), Arrival_Hour (not known at booking time—depends on actual flight), and Arrival_Minute (same issue as Arrival_Hour).

These columns were dropped from the feature set. The remaining features for modeling included: Airline, Source, Destination, Additional_Info (categorical), and Duration_Minutes, Stops, Journey_Day, Journey_Month, Journey_Day_of_Week, Dep_Hour, Dep_Minute (numerical).

5.3 Preprocessing Pipeline

We constructed a scikit-learn ColumnTransformer to handle different feature types appropriately. Categorical features (Airline, Source, Destination, Additional_Info) were encoded using OneHotEncoder with drop='first' to avoid the dummy variable trap. This expanded the 4 categorical columns into 23 binary features.

For numerical features, Duration_Minutes was standardized using StandardScaler to have zero mean and unit variance, which helps algorithms that are sensitive to feature scales. The remaining numerical features (Stops, Journey_Day, Journey_Month, Journey_Day_of_Week, Dep_Hour, Dep_Minute) were passed through unchanged as they were already on reasonable scales.

The final preprocessed feature matrix contained 30 columns: 1 scaled (Duration_Minutes), 6 passthrough (other numericals), and 23 one-hot encoded (categoricals). Critically, the preprocessor was fit only on training data and then applied to both train and test sets to prevent any information leakage from the test set.

6. Phase 5: Model Training & Evaluation

6.1 Model Selection Strategy

Phase 5 evaluated seven different regression algorithms to identify the best performer for this prediction task. The models ranged from simple linear approaches to complex ensemble methods, allowing us to understand the relationship between model complexity and performance.

The baseline models included Linear Regression (simplest possible model, assumes linear relationships) and Ridge Regression (linear model with L2 regularization to prevent overfitting). These established the performance floor and helped us understand how much improvement non-linear models could provide.

The advanced models included Random Forest (ensemble of decision trees with bagging), Gradient Boosting (sequential ensemble that corrects previous errors), and XGBoost (optimized gradient boosting with regularization). We tested both default hyperparameters and tuned versions for Random Forest and XGBoost.

6.2 Model Comparison Results

Model	Test R^2	RMSE	MAE	MAPE
XGBoost (Tuned)	**0.9167**	₹1,163	**₹659**	7.95%
XGBoost (Default)	0.9153	₹1,172	₹706	8.58%
Random Forest (Tuned)	0.9071	₹1,228	₹653	7.82%
Gradient Boosting	0.8286	₹1,668	₹1,186	14.81%
Linear Regression	0.7030	₹2,196	₹1,632	20.77%

The results clearly demonstrate the superiority of ensemble methods over linear approaches for this problem. XGBoost achieved 91.7% R^2, meaning it explains nearly 92% of the variance in flight prices. In contrast, Linear Regression achieved only 70.3% R^2—a 30% improvement from using XGBoost.

The practical difference is even more striking when looking at error metrics. XGBoost's MAE of ₹659 means predictions are off by about ₹660 on average, while Linear Regression's MAE of ₹1,632 represents errors nearly 2.5 times larger. For a ₹8,000 flight, XGBoost predicts within ±8% while Linear Regression is off by ±20%.

6.3 Hyperparameter Tuning

For the top performers (XGBoost and Random Forest), we conducted hyperparameter tuning using RandomizedSearchCV with 5-fold cross-validation. For XGBoost, we explored n_estimators (100, 200, 300), max_depth (3, 5, 7, 10), learning_rate (0.01, 0.05, 0.1, 0.2), and subsample (0.8, 0.9, 1.0).

The optimal XGBoost configuration was: n_estimators=200, max_depth=7, learning_rate=0.1, subsample=0.9, colsample_bytree=0.9. This tuned model achieved a small but meaningful improvement over the default configuration (R^2 0.9167 vs 0.9153).

6.4 Feature Importance Analysis

XGBoost provides feature importance scores indicating each feature's contribution to predictions. The analysis revealed that airline selection is by far the most important factor, with

Airline_Jet Airways alone accounting for 21.93% of the model's predictive power. This confirms that brand and service tier are primary price drivers.

Route information came next: Destination_Delhi (11.96%), Source_Mumbai (7.17%), and Source_Delhi (6.63%). The Additional_Info feature indicating no in-flight meal ranked third at 9.41%, showing that service inclusions significantly impact pricing.

Interestingly, Duration_Minutes and Stops—while important in the exploratory analysis—ranked lower in the model, suggesting that their effects are largely captured through their correlation with other features like airline and route.

7. Phase 6: Deployment

7.1 Modular Code Architecture

The final phase transformed the development notebooks into a production-ready modular codebase. The project was restructured into a proper Python package with separate modules for different functionalities, making the code maintainable, testable, and reusable.

The utils.py module contains constants (airline lists, city lists, duration thresholds), helper functions (parse_duration, parse_stops, is_valid_duration), and validation logic. The data_pipeline.py module provides the DataPipeline class handling data loading, cleaning, feature engineering, and validation. The model.py module defines the FlightPriceModel class with model initialization, preprocessor building, and feature importance extraction.

The train.py module contains the ModelTrainer class managing train-test splitting, model fitting, evaluation, cross-validation, and artifact saving. The inference.py module provides the FlightPricePredictor class for loading saved models and making predictions on new data. Finally, main.py serves as the entry point to run the complete pipeline from command line.

7.2 Streamlit Web Application

A user-friendly web application was built using Streamlit, allowing non-technical users to get flight price predictions. The application presents an intuitive interface where users select their airline, source city, destination, number of stops, journey date, departure time, and flight duration.

The app includes input validation to ensure realistic values—for example, it warns users if they enter a duration that's too short for the selected number of stops (using the same thresholds developed during data cleaning). Upon clicking the predict button, the app displays the predicted price along with a confidence range based on the model's MAE (±₹659).

Additional features include a sidebar showing model performance metrics, duration guidelines for different stop counts, and a tips section with insights from the analysis (e.g., 'Book non-stop flights for potentially lower prices'). The app also displays a summary table of the entered flight details alongside the prediction.

7.3 Saved Artifacts

The deployment package includes all necessary artifacts for making predictions without retraining: best_model.pkl (serialized XGBoost model), preprocessor.pkl (fitted ColumnTransformer for consistent feature transformation), and feature_names.pkl (list of feature names after preprocessing). These files, along with the application code, can be deployed to Streamlit Cloud or any Python hosting environment.

8. Key Findings & Insights

8.1 Counterintuitive Discovery: Non-Stop Flights Are Cheaper

Perhaps the most significant finding from this analysis challenges conventional wisdom about flight pricing. Many travelers assume that non-stop flights command a premium due to their convenience, but our data reveals the opposite: non-stop flights average ₹4,999 while one-stop flights average ₹9,471—making connecting flights 89% more expensive on average.

The explanation lies in understanding what drives operating costs. Connecting flights have longer total durations (more time in the air means more fuel, longer crew duty hours), additional landing and takeoff operations (the most fuel-intensive phases of flight), airport fees at intermediate stops, and more complex logistics. These costs are passed on to passengers, resulting in higher ticket prices despite the inconvenience of connections.

8.2 Airline Brand as Primary Price Driver

The feature importance analysis confirmed that airline selection is the single most important factor in determining flight prices. Jet Airways alone accounts for nearly 22% of the model's predictive power. This reflects the strong brand-based pricing in Indian aviation, where premium carriers can charge significantly more than budget airlines for similar routes.

The practical implication is clear: travelers prioritizing cost savings should focus first on airline choice. Switching from Jet Airways to SpiceJet on the same route could save up to 65% (₹13,097 vs ₹4,479), dwarfing the impact of timing or other factors.

8.3 Temporal Patterns

Several temporal patterns emerged that can help travelers find better deals. Flying on Friday instead of Sunday saves approximately 10%. Departing during red-eye hours (midnight to 4 AM) offers the lowest prices. Booking flights in April rather than May can yield meaningful savings as May marks the beginning of peak summer travel.

8.4 Data Quality Lesson

The discovery of impossible duration values (5-minute connecting flights) highlights the importance of domain knowledge in data cleaning. Standard statistical outlier detection methods would not have caught these errors since the durations weren't extreme in absolute terms. Only by understanding that connecting flights physically require minimum times could we identify and remove these invalid records.

9. Conclusion

9.1 Project Achievements

This project successfully delivered an end-to-end machine learning solution for flight price prediction. Starting from raw, messy data, we developed a comprehensive pipeline that cleans, transforms, and prepares data for modeling. The final XGBoost model achieves 91.7% accuracy with an average prediction error of just ₹659, making it reliable for practical price estimation.

Beyond the predictive model, the project generated actionable business insights that can help travelers make more informed booking decisions. The counterintuitive finding about non-stop flight pricing, the clear airline tier structure, and the temporal pricing patterns all provide value beyond the model itself.

The modular codebase and Streamlit application demonstrate how data science projects can be packaged for real-world deployment. The code is structured for maintainability, with clear separation of concerns and comprehensive documentation.

9.2 Future Enhancements

Several enhancements could further improve this system. Integration with real-time airline APIs would enable dynamic pricing predictions based on current market conditions. Adding booking lead time as a feature (days until departure) could capture the well-known pattern of prices increasing closer to travel dates.

Expanding the model to cover international routes and additional airlines would increase its utility. Implementing price trend forecasting could help users identify optimal booking windows. Finally, adding confidence intervals and uncertainty quantification would provide users with a clearer understanding of prediction reliability.

— End of Report —