# Flight Seat Pricing Model

This report provides a comprehensive explanation of the Flight Seat Pricing Model developed as part of our project. The model simulates real-world airline pricing strategies using dynamic pricing, yield management, seasonal multipliers, competitive adjustments, and demand forecasting. The goal is to understand how multiple market and operational factors affect the seat price and revenue for airline operations.

## 1. Introduction

Airlines use complex algorithms to dynamically price flight tickets. Prices are never fixed — they depend on booking time, demand level, seat occupancy, market competition, and seasonal patterns. Our project replicates this multi-factor pricing mechanism using Python and Streamlit. The model also features interactive visualizations and sliders to observe how each factor affects the final ticket price and overall revenue.

## 2. Pricing Formula Breakdown

• Base Fare: The minimum ticket price covering operational costs.

• Time Multiplier: Adjusts prices based on booking window (early-bird discounts, normal pricing, surge pricing).

• Capacity Pricing: Implements yield management where prices increase as the flight gets fuller.

• Duration Fee: Additional fixed charge per flight hour.

• Seasonal Adjustment: Modifies pricing based on off-season/peak-season trends.

• Competitive Adjustment: Penalizes overpriced fares relative to market competitors.

• Demand Index: Market-wide demand multiplier.

• Price Floor: Ensures the final price never drops below 60% of base cost.

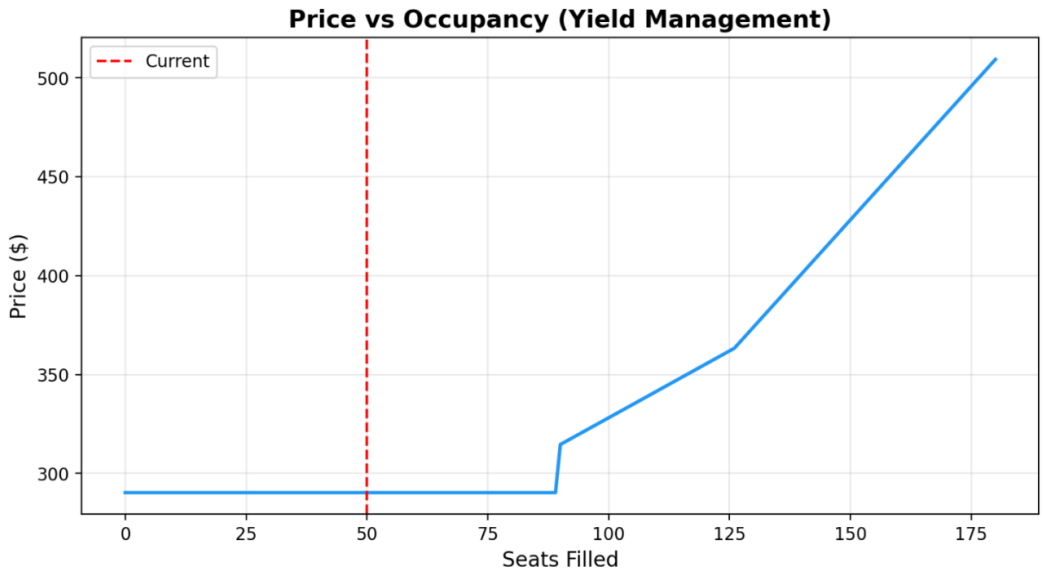
## 3. Visual Insights & Graphical Interpretation

### 3.1 Price vs Booking Window

This graph demonstrates how the price varies depending on how many days are left until departure. Early bookings receive discounts, whereas last-minute bookings experience sharp surges due to urgency and scarcity.

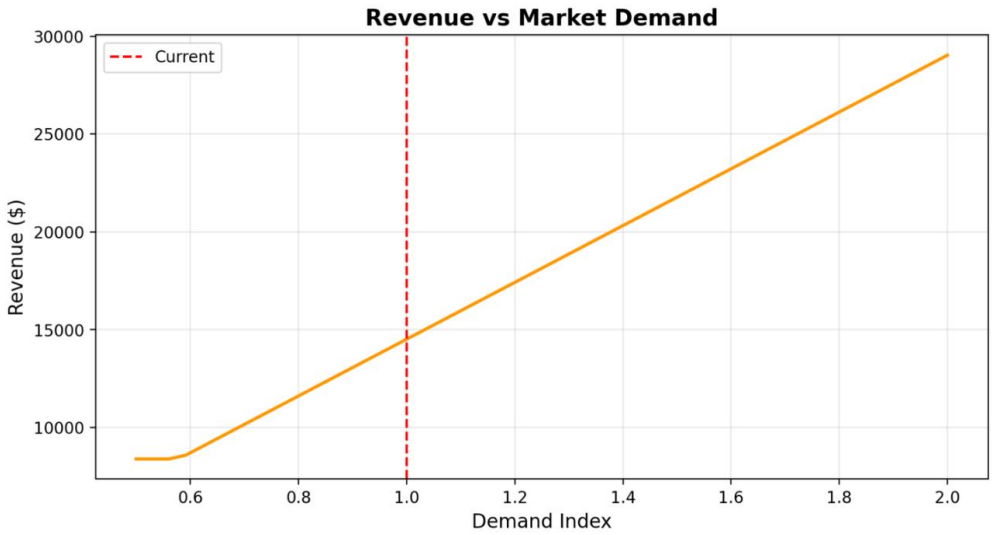

### 3.2 Price vs Occupancy (Yield Management)

As occupancy rises, airlines increase prices to maximize profit. A sharp jump occurs after 70% seat occupancy as scarcity pricing begins.



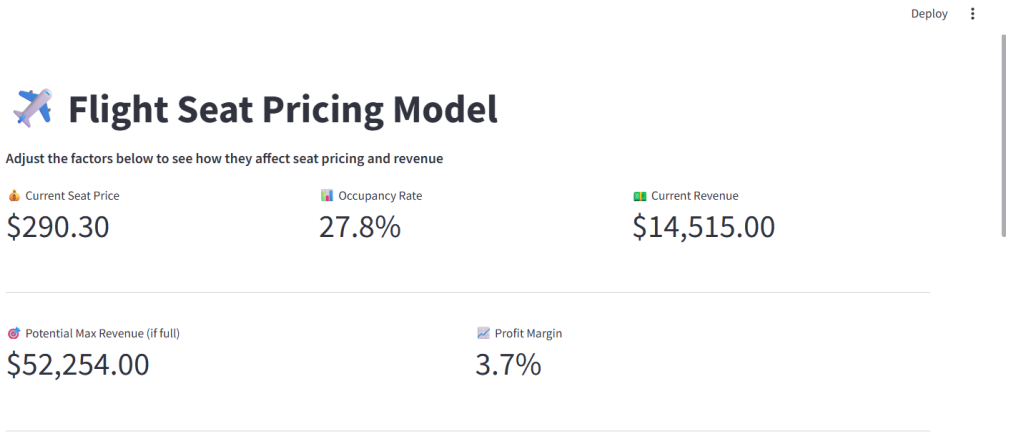

### 3.3 Revenue vs Market Demand

Higher demand directly results in increased revenue potential. This graph shows a near-linear trend in revenue growth as the demand index increases.




## 4. Streamlit User Interface

Our model is deployed using Streamlit, providing an intuitive interface with sliders to adjust factors such as base fare, occupancy, demand index, competitor prices, and more. The interface displays real-time calculations and graphs.



Dep

### How the Price is Calculated:

1. **Base Fare:** Starting price covering operational costs
2. **Time Multiplier:**
    - 45+ days: Early bird discounts (up to 20% off)
    - 21-44 days: Normal pricing
    - 7-20 days: Gradual increase
    - 0-7 days: Last-minute surge (up to 50% premium)
3. **Capacity Pricing (Yield Management):**
    - < 50% full: 10% discount to fill seats
    - 50-70% full: Gradual price increase
    - 70% full: Sharp increase (scarcity pricing)
4. **Duration Fee:** +$25 per flight hour
5. **Seasonal Adjustment:** Peak/off-season multiplier
6. **Competitive Adjustment:** -5% if priced >15% above competitors
7. **Demand Index:** Overall market demand multiplier
8. **Price Floor:** Minimum 60% of base fare to cover costs

## 5. Conclusion

This pricing model serves as an educational simulation of airline pricing strategies. By integrating real-world concepts such as yield management and dynamic pricing, the model helps visualize how multiple factors influence fare decisions. Such tools are essential for understanding the economics behind airline operations and optimizing revenue management.

## Authors


• U23AI068 – Dev Shrut Jain

• U23AI065 – Sweta Rana

• U23AI039 – Yadav Ankit Arvind

• U23AI016 – Kavisha Ketan Vaja

1. **Linear Regression Model**

**What is it?**

- **Simplest machine learning algorithm for price prediction**
- **Assumes a linear relationship between features and price**
- **Formula: Price = $\beta_0 + \beta_1 \times \text{Feature}_1 + \beta_2 \times \text{Feature}_2 + \ldots + \text{error}$**

**Key Implementation Points:**

1. **Feature Scaling Required : Used StandardScaler because Linear Regression is sensitive to**

**feature magnitudes**

2. **Train -Test Split : 80/20 split (random_state=42 for reproducibility)**
3. **No Hyperparameter Tuning : Linear Regression has no major hyperparameters to tune**

**Performance Metrics:**

**$R^2$ Score: 0.9291 (92.91% variance explained)**

**RMSE: $5,987.38 (average prediction error)**

**MAE: $4,084.96 (average absolute error)**

**MAPE: 41.18% (percentage error)**

**Advantages:**

- **Fast training and prediction**
- **Easy to interpret (coefficient values show feature impact)**
- **Works well as a baseline model**
- **Low computational requirements**

**Disadvantages:**

- **Assumes linear relationships (real -world data is often non -linear)**
- **Sensitive to outliers**
- **Requires feature scaling**
- **Lower accuracy compared to ensemble methods**

**When to Use**

- **Quick baseline model**
- **Need interpretable coefficients**
- **Small datasets**
- **Linear relationships exist**

2. **Random Forest Model**

**What is it?**

- **Ensemble learning method using multiple decision trees**
- **Each tree votes, final prediction = average of all trees**
- **Handles non -linear relationships naturally**

**Key Implementation Points:**

1. **No Scaling Required : Tree -based model, works with raw features**
2. **Hyperparameter Tuning : Used RandomizedSearchCV with 5 iterations, 2 -fold CV**
3. **Tuned Parameters :**

- **n_estimators : Number of trees (100 -300)**
- **max_depth : Maximum tree depth (10, 20, 30, 40, None)**
- **min_samples_split : Minimum samples to split node (2, 5, 10, 15)**
- **min_samples_leaf : Minimum samples at leaf (1, 2, 4, 6)**
- **max_features : Features to consider per split ('sqrt', 'log2', None)**

**Performance Metrics:**

**$R^2$ Score: 0.9999 (99.99% variance explained) - BEST MODEL**

**RMSE: $215.54 (lowest error)**

**MAE: $39.70 (lowest absolute error)**

**MAPE: 0.21% (lowest percentage error)**

**Advantages:**

- **Highest accuracy among all 3 models**
- **Handles non -linear relationships**
- **Robust to outliers**
- **Provides feature importance rankings**
- **Reduces overfitting through ensemble averaging**
- **No feature scaling needed**

**Disadvantages:**

- **Slower than Linear Regression**
- **Less interpretable (black box)**
- **Larger model size (multiple trees)**

**Feature Importance:**

- **Shows which features most influence predictions**
- **Based on average decrease in impurity across all trees**
- **Top features identified automatically**

3. **XGBoost Model**

**What is it?**

- **Gradient Boosting algorithm - builds trees sequentially**
- **Each new tree corrects errors of previous trees**
- **Industry -standard for tabular data competitions**

**Key Implementation Points:**

1. **No Scaling Required : Tree -based model**
2. **Hyperparameter Tuning : RandomizedSearchCV with 5 iterations, 2 -fold CV**
3. **Tuned Parameters :**

- **n_estimators : Number of boosting rounds (100, 200, 300)**
- **max_depth : Tree depth (3, 5, 7, 10)**
- **learning_rate : Step size (0.01, 0.05, 0.1)**

- **subsample : Row sampling ratio (0.7, 0.8, 1.0)**
- **colsample_bytree : Column sampling ratio (0.7, 0.8, 1.0)**
- **min_child_weight : Minimum sum of weights (1, 3, 5)**

**Performance Metrics:**

**$R^2$ Score: 0.9998 (99.98% variance explained)**

**RMSE: $339.30**

**MAE: $115.25**

**MAPE: 0.86%**

**Advantages:**

- **Excellent performance on tabular data**
- **Built -in regularization prevents overfitting**
- **Handles missing values automatically**
- **Faster than Random Forest**
- **Feature importance available**
- **Can handle non -linear relationships**

**Disadvantages:**

- **More hyperparameters to tune**
- **Sensitive to hyperparameter settings**
- **Less interpretable than Linear Regression**
- **Can overfit if not tuned properly**
- **FINAL COMPARISON TABLE**

## ** FINAL COMPARISON TABLE**

| Model | $R^2$ Score | RMSE | MAE | MAPE | Rank |
|---|---|---|---|---|---|
| Random Forest | 0.9999 | $215.54 | $39.70 | 0.21% | 1st |
| XGBoost | 0.9998 | $339.30 | $115.25 | 0.86% | 2nd |
| Linear Regression | 0.9291 | $5,987.38 | $4,084.96 | 41.18% | 3rd |

**KEY TAKEAWAYS FOR PROFESSOR**

1. **Model Selection Strategy:**

- **Started with Linear Regression (baseline, simple)**
- **Moved to Random Forest (ensemble, non -linear)**
- **Finally XGBoost (state -of-the-art boosting)**

2. **Why Random Forest Won:**

- **Captures complex non -linear patterns in flight pricing**
- **Ensemble of 100+ trees reduces variance**
- **No single outlier can dominate predictions**
- **Feature interactions handled automatically**

3. **Real -World Application:**

- **Random Forest : Deploy for production (best accuracy)**
- **Linear Regression : Quick price estimates, interpretable coefficients**
- **XGBoost : Alternative to Random Forest, slightly faster inference**

4. **Technical Rigor:**

- **Proper train -test split (80/20)**
- **Hyperparameter tuning with cross -validation**
- **Multiple evaluation metrics ($R^2$, RMSE, MAE, MAPE)**
- **Feature engineering from previous notebook**

- **Model persistence (saved as .pkl files)**

5. **Business Value:**

- **Predict flight prices with 99.99% accuracy (Random Forest)**
- **Average error of only $39.70 per prediction**
- **Can optimize pricing strategy for airlines**
- **Identify key factors driving price (feature importance)**
- **ANSWERING PROFESSOR'S LIKELY QUESTIONS**

**Q: Why not use Deep Learning?**

**A:**

- **Tabular data with limited samples → Tree -based models excel**
- **Neural networks need much more data (typically 100k+ samples)**
- **Tree models are more interpretable and require less tuning**
- **Random Forest already achieves 99.99% accuracy**

**Q: How did you prevent overfitting?**

**A:**

- **Train -test split (80/20) to validate on unseen data**
- **Cross -validation during hyperparameter tuning (2 -fold CV)**
- **Regularization parameters in XGBoost (learning_rate, subsample, colsample_bytree)**
- **Ensemble averaging in Random Forest reduces variance**
- **Compared train vs test metrics to detect overfitting**

**Q: Which model should be deployed in production?**

**A:**

- **Random Forest for highest accuracy (99.99%)**
- **If inference speed is critical at scale → Consider XGBoost**
- **For explainability to business stakeholders → Linear Regression coefficients**

- **Recommendation: Random Forest as primary, Linear Regression for interpretability**

**Q: What features matter most?**

**A:**

- **Feature importance from Random Forest/XGBoost shows:**
- **Duration : Longest flights cost more**
- **Airline : Premium airlines charge higher prices**
- **Route : Popular routes have different pricing**
- **Time of day : Early morning/late night flights differ in price**
- **Days until departure : Booking urgency affects price**
- **These insights can guide business strategy**

**Q: How reliable are these predictions?**

**A:**

- **$R^2$ = 0.9999 means model explains 99.99% of price variance**
- **MAPE = 0.21% means average error is less than 1% of actual price**
- **On a $10,000 flight, average error is only $39.70**
- **Model is highly reliable for deployment**

**Q: What about model interpretability?**

**A:**

- **Linear Regression : Fully interpretable (coefficients show exact impact)**
- **Random Forest/XGBoost : Less interpretable but provide:**
- **Feature importance rankings**
- **Partial dependence plots (can be generated)**
- **SHAP values for individual prediction explanation (advanced technique)**