INNOMATİCS
RESEARCH LABS

Machine Learning - Project Report Document

Student Name	Sada Vijay
Batch	AI Elite 18
Project Name	Flight Ticket Price Prediction
Project Domain	Predictive analytics
Type of Machine Learning	Supervised ML
Type of Problem	Regression
Project Methodology	CRISP-DM
Stages Involved	• Data Collection and Understanding • Data Preparation • Model Building • Model Training • Model Evaluation • Model Deployment

Business Understanding:

The concept of flight price prediction involves analyzing historical data and various factors that influence the cost of airline tickets, such as the airline, departure date, booking class, and other relevant features. The goal is to predict the fare of a flight ticket for customers, enabling them to make informed decisions and potentially save money on their travel expenses.

The airline industry is highly competitive and dynamic, with ticket prices fluctuating based on demand, seasonality, fuel prices, and other factors. By leveraging predictive models, airlines and travel agencies can optimize pricing strategies, enhance customer satisfaction, and increase revenue.

In this context, flight price prediction serves multiple purposes:

- **Customer Decision-Making:** Helping customers find the best deals and plan their travel budget effectively.
- **Revenue Management:** Assisting airlines in setting competitive prices while maximizing revenue.
- **Market Analysis:** Providing insights into market trends and customer preferences.

Problem Statement:

This project aims to develop a model that can predict the fare of a flight ticket based on features such as airline, departure date, booking class, and other relevant factors. This will enable more accurate pricing strategies and enhance customer satisfaction by providing fare predictions.

Here are some potential business constraints:

- Regulatory Compliance: Ensuring the model adheres to aviation industry regulations and pricing guidelines.
- Data Privacy and Security: Protecting customer data and maintaining privacy in compliance with relevant laws.
- Resource Limitations: Managing computational resources and time constraints for model training and deployment.
- Accuracy and Reliability: Striving for high accuracy in predictions while maintaining model reliability.
- Interpretability: Balancing the complexity and interpretability of the model to ensure stakeholders understand pricing variations.
- Ethical Considerations: Ensuring fair pricing practices and avoiding any discriminatory pricing based on customer profiles.
- Competitive Landscape: Staying competitive by adapting to market trends and competitor pricing strategies.

Stage 1: Data Collection and Understanding

a) **Data Collection:** The dataset source for this project was Kaggle. The Data was collected in two parts(datasets): one for economy class tickets and another for business class tickets. A total of 300261 distinct flight booking options was extracted from the website Easemytrip for flight travel between India's top 6 metro cities. The data contains information on flights for 50 days, from February 11th to March 31st, 2022. The both datasets were later combined into a single dataset after doing cleaning.

b) **Data Understanding:**

1. Airline: The name of the airline company. It is a categorical feature having 6 different airlines.
2. Flight: Information regarding the plane's flight code. It is a categorical feature.
3. Source City: The city from which the flight takes off. It is a categorical feature having 6 unique cities.
4. Departure Time: A derived categorical feature created by grouping time periods into bins. It stores information about the departure time and has 4 unique time labels.
5. Stops: A categorical feature with 3 distinct values that stores the number of stops between the source and destination cities.
6. Arrival Time: A derived categorical feature created by grouping time intervals into bins. It has 4 distinct time labels and keeps information about the arrival time.
7. Destination City: The city where the flight will land. It is a categorical feature having 6 unique cities.
8. Class: A categorical feature that contains information on seat class; it has two distinct values: Business and Economy.
9. Duration: A continuous feature that displays the overall amount of time it takes to travel between cities in hours.
10. Days Left: A derived characteristic calculated by subtracting the trip date from the booking date.
11. Price: The target variable that stores information about the ticket price.

S No	Feature Name	Data Type
1	Date	Object
2	Airline	Object
3	Flight	Object
4	From	Object
5	Departure time	Object
6	Stops	Object
7	Destination	Object

8	Arrival time	Object
9	Class	Object
10	Duration	Float64
11	Days left	Int64
12	Price	Float64

Stage 2: Data Preparation

a) Exploratory Data Analysis:

S No	Type	Feature Names	Observation
1	Missing Values	NA	NA
2	Duplicates	All columns	There exist 3195 duplicate datapoints in the dataset.
3	Outliers	Duration, Price	There exist outliers in these columns.

b) Data Cleaning/wrangling:

S no	Type of Cleaning	Technique	Feature Name	Reason
1	Duplicate value	Drop	All columns	To maintain data consistency and accuracy as they don't carry any useful information. Dropped 3195 datapoints.
2	Encoding	Binary Encoder	Airline, From. Departure time, Stops, Destination, Arrival time, Class	Used Binary Encoding since the data in these categorical columns are nominal with a high cardinality ranging from 4 to 8.
3	Scaling	Robust Scaling	Duration, Days left	Used Robust Scaling since there exists outliers in the column "Duration" and robust scaler handles the outliers better. Since they contain true outliers.

c) Feature Selection:

S no	Removed Feature Name	Reason	Test Performed
1	Date	Dropped this column since we've created a replacement for this column named 'days left'	NA
2	Flight	Dropped this column since it has high cardinality making it similar to IDs type of data.	NA
3	From	Dropped because of low feature importance in all observations.	Lasso, DecisionTreeRegressor, RandomForestRegressor
4	Destination	Dropped because of low feature importance in all observations.	Lasso, DecisionTreeRegressor, RandomForestRegressor
5	Departure time	Dropped because of low feature importance in all observations.	Lasso, DecisionTreeRegressor, RandomForestRegressor
6	Arrival time	Dropped because of low feature importance in all observations.	Lasso, DecisionTreeRegressor, RandomForestRegressor

Stage 3: Model Building:

S No	Type of Problem	Approach	Algorithm Name
1	Regression	Distance-Based	KNeighborsRegressor
2	Regression	Decision Tree	DecisionTreeRegressor
3	Regression	Linear Model	LinearRegression
4	Regression	Robust Linear Model	RANSACRegressor
5	Regression	Robust Linear Model	TheilSenRegressor
6	Regression	Robust Linear Model	HuberRegressor
7	Regression	Linear Model with Regularization	Lasso
8	Regression	Linear Model with Regularization	Ridge
9	Regression	Linear Model with Regularization	ElasticNet
10	Regression	Ensemble - Bagging	RandomForestRegressor
11	Regression	Ensemble - Boosting	GradientBoostingRegressor
12	Regression	Ensemble - Boosting	XGBRegressor
13	Regression	Ensemble - Boosting	AdaBoostRegressor

1. **KNeighbors Regressor:** K-nearest neighbors (KNN) regression predicts the target variable by averaging the values of its k-nearest neighbors in the feature space. It assumes that similar data points have similar target values, making it suitable for locally smooth relationships between features and the target.
2. **Decision Tree Regressor:** Decision tree regression builds a model that predicts the target variable by partitioning the data into subsets based on the values of input features. It recursively splits the data based on feature thresholds, aiming to minimize the variance of the target variable within each subset.
3. **Linear Regression:** Linear regression models the relationship between the dependent variable and one or more independent variables by fitting a linear equation. It assumes a linear relationship between the variables and is widely used for predicting continuous outcomes.
4. **RANSAC Regressor:** RANSAC (RANdom SAmple Consensus) regression fits a regression model to a subset of data points (inliers) while ignoring outliers. It iteratively refits the model to improve accuracy by minimizing the impact of outliers on the model coefficients.

5. **Theil-Sen Regressor:** Theil-Sen regression estimates the slope of the relationship between variables using the median of slopes between all pairs of sample points. It is robust to outliers and works well in the presence of noise and heteroscedasticity (unequal variance across data).
6. **Huber Regressor:** Huber regression combines the best properties of least squares and least absolute deviation methods. It minimizes the sum of squared errors for samples close to the regression line (like least squares) and absolute error for samples far from it (like least absolute deviation).
7. **Lasso Regression:** Lasso (Least Absolute Shrinkage and Selection Operator) regression adds a penalty to the sum of absolute values of the regression coefficients, promoting sparsity and feature selection by shrinking some coefficients to zero.
8. **Ridge Regression:** Ridge regression adds a penalty to the sum of squared coefficients (L2 regularization), reducing the effect of multicollinearity and shrinking the coefficients towards zero, but rarely to zero.
9. **ElasticNet Regression:** ElasticNet regression combines penalties from both Lasso and Ridge, using a convex combination of L1 and L2 regularization terms. It balances between feature selection (like Lasso) and handling multicollinearity (like Ridge).
10. **Random Forest Regressor:** Random forest builds multiple decision trees during training and outputs the average prediction of the individual trees. It reduces overfitting compared to a single decision tree and provides high accuracy.
11. **Gradient Boosting Regressor:** Gradient boosting builds an ensemble of trees sequentially, where each tree corrects the errors of the previous one. It combines the predictions of multiple weak learners (decision trees) to produce a strong prediction model.
12. **XGBoost Regressor:** XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library designed for efficient computation. It improves upon traditional gradient boosting with system optimizations and algorithmic enhancements.
13. **AdaBoost Regressor:** AdaBoost (Adaptive Boosting) combines multiple weak learners (typically decision trees) to create a strong predictor. It assigns higher weights to incorrectly predicted instances, focusing subsequent learners on harder cases.

Stage 4: Model Training:

Basic Models:

Model	Train MAE	Train MSE	Train RMSE	Train R2	Train Adj R2	Test MAE	Test MSE	Test RMSE	Test R2	Test Adj R2
KNeighbours	2791.82	2.28 E+07	4777.5	0.96	0.96	3427.56	3.49 E+07	5904.78	0.93	0.93
DecisionTree	1033.68	7.52 E+06	2743.07	0.99	0.99	3874.28	4.92 E+07	7013.73	0.91	0.91
LinearRegression	4592.36	4.84 E+07	6957.66	0.91	0.91	4594.61	4.94 E+07	7030.13	0.91	0.91
RANSAC	4544.57	6.18 E+07	7861.98	0.88	0.88	4561.98	6.27 E+07	7916.18	0.88	0.88
TheilSen	4477.81	4.91 E+07	7005.85	0.9	0.9	4491.58	5.02 E+07	7084.13	0.9	0.9
HuberRegressor	4270.72	5.15 E+07	7173.77	0.9	0.9	4288.27	5.27 E+07	7259.65	0.9	0.9
Lasso	4593.04	4.84 E+07	6957.96	0.91	0.91	4596.02	4.94 E+07	7030.97	0.91	0.91
Ridge	4593.41	4.84 E+07	6957.65	0.91	0.91	4595.61	4.94 E+07	7030.34	0.91	0.91
Elastic Net	10524.03	1.73 E+08	13140.6	0.66	0.66	10685.08	1.78 E+08	13353	0.66	0.66
RandomForest	1679.38	9.89 E+06	3144.75	0.98	0.98	3341.12	3.49 E+07	5906.82	0.93	0.93
GradientBoosting	3175.06	2.91 E+07	5393.16	0.94	0.94	3196	2.95 E+07	5434.52	0.94	0.94
XGBoost	2779.56	2.25 E+07	4742.14	0.96	0.96	3059.9	2.80 E+07	5294.74	0.95	0.95
AdaBoost	3726.34	3.48 E+07	5901.17	0.93	0.93	3737.32	3.54 E+07	5946.64	0.93	0.93

- From above we can observe that Elastic Net has quite the poor performance of all the models and XGBoost has the best overall R2 scores on both train and test data.

Hyper-Parameter Tuning Using RandomizedSearchCV:

S No	Algorithm Name	Hyper-parameter tuning	Metric used for Evaluation
1	KNN	n_neighbors: 2-30, weights: uniform, distance	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
2	Decision Tree	max_depth: 1-45 (step 5)	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
3	Linear Regression	None	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
4	RANSAC Regression	None	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
5	TheilSen Regression	None	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
6	Huber Regression	None	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
7	Lasso Regression	alpha: 0.01, 0.1, 1, 10, 100	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
8	Ridge Regression	alpha: 0.01, 0.1, 1, 10, 100	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
9	Random Forest	n_estimators: 50-199, max_depth: None, 10, 20, 30, 40, 50	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
10	Gradient Boosting	n_estimators: 50-199, max_depth: 3, 4, 5, 6, None	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
11	XGBoost	n_estimators: 50-199, max_depth: 3, 4, 5, 6, 8, 10, learning_rate: 0.01, 0.1, 0.15, 0.3	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2
12	AdaBoost	n_estimators: 50-199, learning_rate: 0.01, 0.1, 0.15, 0.3	Mean Squared Error (MSE), R-squared, MAE, RMSE, ADJ_R2

Stage 5: Model Evaluation:

S No	Model	Training Time	Testing Time	R2 Score	Adj R2 Score	Hyperparameter
1	KNeighbours	0.07761	1.048659	0.94	0.94	n_neighbors=29
2	DecisionTree	0.072904	0.004241	0.94	0.94	max_depth=11
3	LinearRegression	0.01687	0.004502	0.91	0.91	None
4	RANSAC	0.321958	0.002916	0.87	0.87	None
5	TheilSen	12.76159	0.004496	0.90	0.90	None
6	HuberRegressor	0.501576	0.002852	0.90	0.90	None
7	Lasso	0.033868	0.002928	0.91	0.91	alpha=0.01
8	Ridge	0.00771	0.002706	0.91	0.91	alpha=0.1
9	RandomForest	3.781979	0.123334	0.95	0.95	max_depth=10, n_estimators=79
10	GradientBoosting	8.884084	0.071349	0.95	0.95	max_depth=6, n_estimators=160
11	XGBoost	0.24346	0.021409	0.95	0.95	max_depth=5, n_estimators=74, learning_rate=0.3
12	AdaBoost	3.170216	0.094205	0.93	0.93	learning_rate=0.1, n_estimators=79

From the above table, here are a few observations:

- RandomForest, GradientBoosting, and XGBoost models achieve the highest R2 and Adjusted R2 Scores of 0.95.
- The TheilSen model has an exceptionally high training time (12.761589 seconds), significantly longer than any other model.
- The KNeighbours model has the highest testing time (1.048659 seconds), which is much longer compared to other models.
- Ridge has the shortest training time (0.007710 seconds) and very short testing time (0.002706 seconds).

- XGBoost balances well with a relatively short training time (0.243460 seconds) and testing time (0.021409 seconds).
- LinearRegression, Ridge, and Lasso models all have identical R2 and Adjusted R2 Scores of 0.91, indicating similar predictive performance.

Stage 6: Model Deployment:

The flight ticket price prediction model has been deployed via a Streamlit web application on a local server environment. This deployment aims to provide users with a convenient interface for predicting flight ticket prices based on various parameters.

Deployment Environment:

The Streamlit app is hosted on a local machine with Python and necessary libraries installed to support the application's functionality.

Deployment Steps:

1. Environment Setup:
   - Installed required Python packages including Streamlit for web application development.
2. Application Development:
   - Developed the Streamlit web application using Python, incorporating the flight ticket price prediction model.
3. Testing and Validation:
   - Conducted testing to ensure the app functions correctly, handling various inputs and scenarios effectively.

Functionality:

The deployed app enables users to input flight details such as airline, class, date, and other relevant parameters. It then provides an estimated price for the flight based on historical data and machine learning predictions.

Usage Instructions:

1. Input Details:
   - Users can enter details such as departure city, destination, departure date, number of passengers, etc.
2. Prediction Output:
   - Upon submitting the details, the app calculates and displays the predicted flight ticket price using the deployed machine learning model.

Future Considerations:

Future enhancements may involve:

- Exploring deployment options on cloud servers for broader accessibility.
- Adding features like real-time updates, multi-city travel predictions, and user feedback mechanisms to improve the application's utility and user experience.

Challenges Faced:

- While cleaning the dataset, some tags which we're causing problems for changing the wrong data type to correct like "stops" and "time taken".
- Also, while dealing with the datetime format columns.
- While doing the hyper-parameter tuning and selections of parameters.

Conclusion:

The best models are the following:

- Decision Tree can be considered the best model with quite the fast prediction time (0.004241 seconds) and with a slight trade-off in the R2 Score (0.94).
- XGBoost has the best overall performance due to its perfect balance of high R2 Score (0.95) and a bit slow prediction time (0.071349 seconds).

Thus, Decision Tree can be considered the best choice for scenarios where computational efficiency is prioritized and a slight reduction in predictive performance (from 0.95 to 0.94) is acceptable.