```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncc
from sklearn.linear_model import LinearRegr
from sklearn.metrics import mean_squared_er
from sklearn.ensemble import RandomForestRe
import warnings
warnings.filterwarnings("ignore")
```

```
dt=pd.read_csv('/content/Clean_Dataset.csv'
```

```
dt.shape
```

```
(300153, 12)
```

```
dt.isnull().values.any() # boolen methoud t
```

```
False
```

```
dt.isnull().sum()
```

```
NO                  0
airline             0
flight              0
source_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
class               0
duration            0
days_left           0
price               0
dtype: int64
```

```
dt = dt.dropna() # to double check
```

```
dt.duplicated(subset=None, keep='first')
```

```
0         False
1         False
2         False
3         False
4         False
          ...
300148    False
300149    False
300150    False
300151    False
300152    False
Length: 300153, dtype: bool
```

```
dt.shape
```

```
(300153, 12)
```

```
dt.info() # checking the data type of every column
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column            Non-Null Count
---  ------            --------------
 0   NO                300153 non-null
 1   airline           300153 non-null
 2   flight            300153 non-null
 3   source_city       300153 non-null
 4   departure_time    300153 non-null
 5   stops             300153 non-null
 6   arrival_time      300153 non-null
 7   destination_city  300153 non-null
 8   class             300153 non-null
 9   duration          300153 non-null
 10  days_left         300153 non-null
 11  price             300153 non-null
dtypes: float64(1), int64(3), object(8)
memory usage: 27.5+ MB
```

Start coding or generate with AI.

```
dt.head()
```

	NO	airline	flight	source_city	dep
0	0	SpiceJet	SG-8709	Delhi	
1	1	SpiceJet	SG-8157	Delhi	
2	2	AirAsia	I5-764	Delhi	
3	3	Vistara	UK-995	Delhi	
4	4	Vistara	UK-963	Delhi	

```
dt.tail()
```

	NO	airline	flight	source_
300148	300148	Vistara	UK-822	Che
300149	300149	Vistara	UK-826	Che
300150	300150	Vistara	UK-832	Che
300151	300151	Vistara	UK-828	Che
300152	300152	Vistara	UK-822	Che

Rose A
6 May 2024
(edited 6 May 2024)

we can see that the min duration 0.83 which is correct becoz some flight take less than hour like 0.83= 50 minutes
so no impure/of low quality

Rose A
6 May 2024

no need to do any update on the data

```
dt.describe() #generate various summary stati
#Note: Only features with numeric data are co
```

	NO	duration	day
count	300153.000000	300153.000000	300153
mean	150076.000000	12.221021	26
std	86646.852011	7.191997	13
min	0.000000	0.830000	1
25%	75038.000000	6.830000	15
50%	150076.000000	11.250000	26
75%	225114.000000	16.170000	38
max	300152.000000	49.830000	49

Rose A
6 May 2024

add a name for the first feature

```
dt.columns = dt.columns.str.replace('Unnamed: 0', 'NO')
```

```
dt.columns #check if the name changed
```

```
Index(['NO', 'airline', 'flight',
'source_city', 'departure_time',
'stops',
       'arrival_time',
'destination_city', 'class',
'duration', 'days_left',
       'price'],
      dtype='object')
```

```
dt.describe(include = 'object') #summary st
```

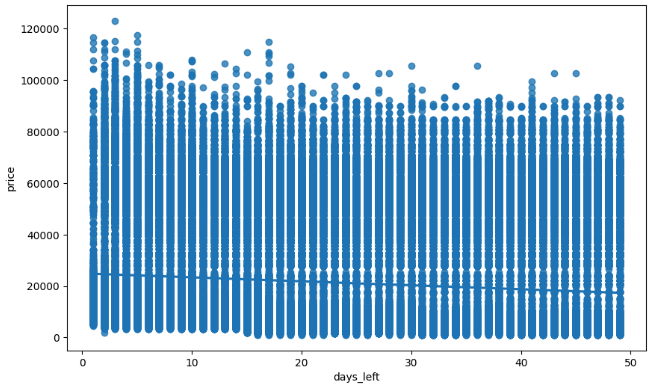
Rose A
6 May 2024

its an inverse relationship as the number increase the price decrease
Moreover :

	airline	flight	source_city	de
count	300153	300153	300153	
unique	6	1561	6	
top	Vistara	UK-706	Delhi	
freq	127859	3235	61343	

```
import seaborn as sns #for making statistic
import matplotlib.pyplot as plt #collectior
plt.figure(figsize=(10,6))
sns.regplot(x="days_left", y="price", data=
```

```
<Axes: xlabel='days_left',
ylabel='price'>
```




```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(dt['
print("The Pearson Correlation Coefficient
```

```
The Pearson Correlation Coefficient is
```

```
plt.figure(figsize=(10,6))
sns.regplot(x="duration", y="price", data=c
```

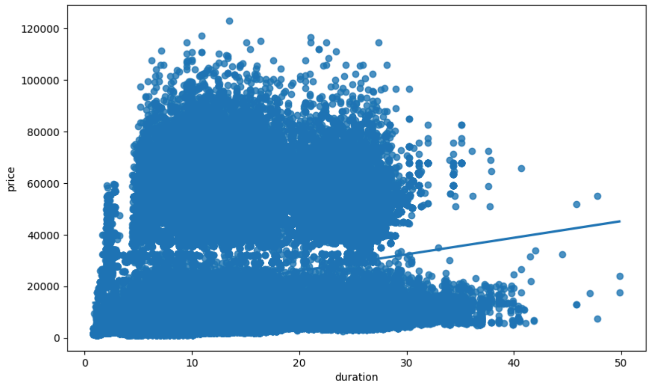
the number of days left before a flight can affect the price. Generally, booking flights well in advance can sometimes result in lower prices, especially for popular routes and peak travel times. However, prices can also fluctuate based on demand, seat availability, airline pricing strategies, and other factors. Last-minute bookings, on the other hand, might result in higher prices due to limited availability and increased demand. It's advisable to monitor prices and book flights when you find a suitable balance between price and convenience.

R Rose A
6 May 2024

As observed in the plot above, a negative correlation is observed

Rose A
6 May 2024

```
<Axes: xlabel='duration',
ylabel='price'>
```



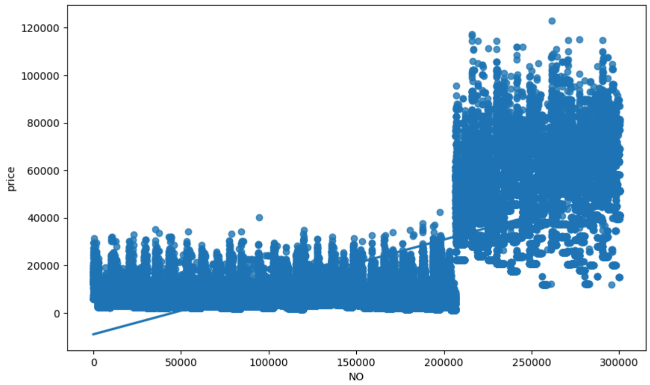
A 0.2 correlation is calculated which is very small with a p value of 0. This indicates that even though the correlation is small but its 20% of 100 which is significant hence this feature can be used for predicition. In addition, p value confirms its importance

```
pearson_coef, p_value = stats.pearsonr(dt['
print("The Pearson Correlation Coefficient
```

```
The Pearson Correlation Coefficient is
```

```
plt.figure(figsize=(10,6))
sns.regplot(x="NO", y="price", data=dt)
```

```
<Axes: xlabel='NO', ylabel='price'>
```



Rose A
6 May 2024

As observed above, a high positive correlation of 0.7 is calculated along with the p-value of 0.Nevertheless , this is just numbers for the data set which can not be at all effecting the predication

```
pearson_coef, p_value = stats.pearsonr(dt['
print("The Pearson Correlation Coefficient
```

```
The Pearson Correlation Coefficient is
```

Rose A
6 May 2024

In the given plot below, it is observed that the price range vary for airline. This indicates the categories can vary with price hence feature can be used for prediction

```
#Box Plot
```

```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column            Non-Null Count
---  ------            --------------
 0   NO                300153 non-null
 1   airline           300153 non-null
 2   flight            300153 non-null
 3   source_city       300153 non-null
 4   departure_time    300153 non-null
 5   stops             300153 non-null
 6   arrival_time      300153 non-null
 7   destination_city  300153 non-null
 8   class             300153 non-null
 9   duration          300153 non-null
 10  days_left         300153 non-null
 11  price             300153 non-null
dtypes: float64(1), int64(3), object(8)
memory usage: 27.5+ MB
```

```
sns.boxplot(x="airline", y="price", data=dt ) #X to metion x-aixs y to met
```

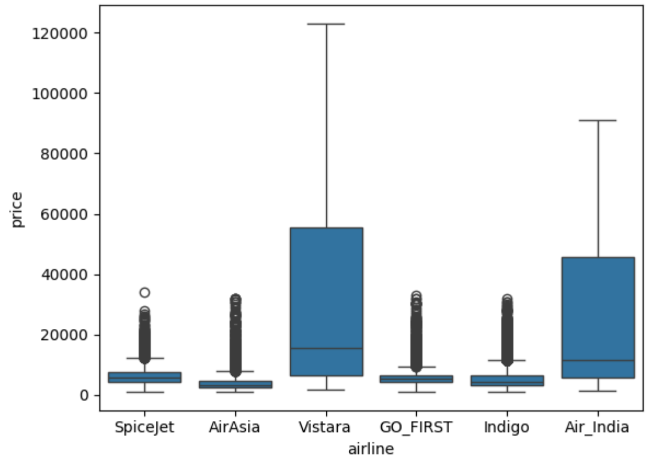
Rose A
6 May 2024
(edited 6 May 2024)

its show the is not affecting

Rose A
6 May 2024

is not that important when selling a ticket. The variety of price ranges for all categories prove that the feature is insignificant for price prediction

```
<Axes: xlabel='airline',
ylabel='price'>
```

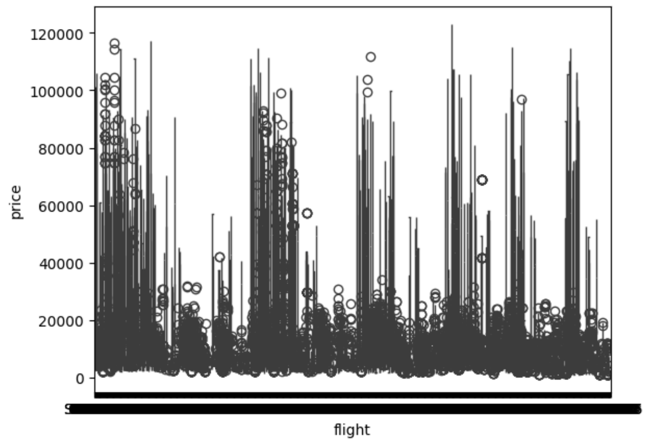


Rose A
6 May 2024

In the given plot below, it is observed that the price range vary for source city. This indicates the categories can vary with price hence feature can be used for prediction

```
sns.boxplot(x="flight", y="price", data=dt )
```

```
<Axes: xlabel='flight',
ylabel='price'>
```

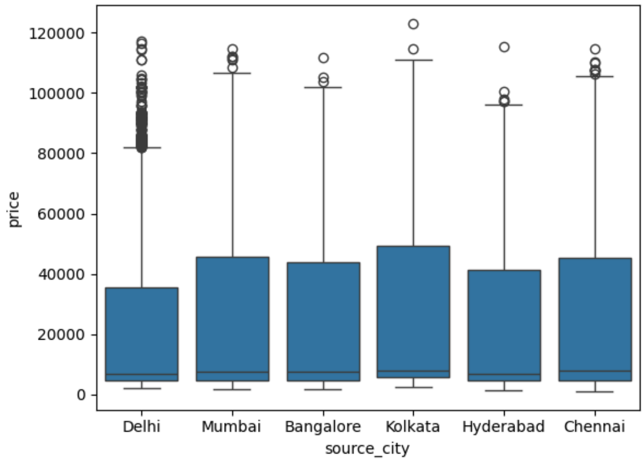


Rose A
19:48 Today

Delhi is less than Kolkata with more than 20000

```
sns.boxplot(x="source_city", y="price", data=dt )
```

```
<Axes: xlabel='source_city', ylabel='price'>
```

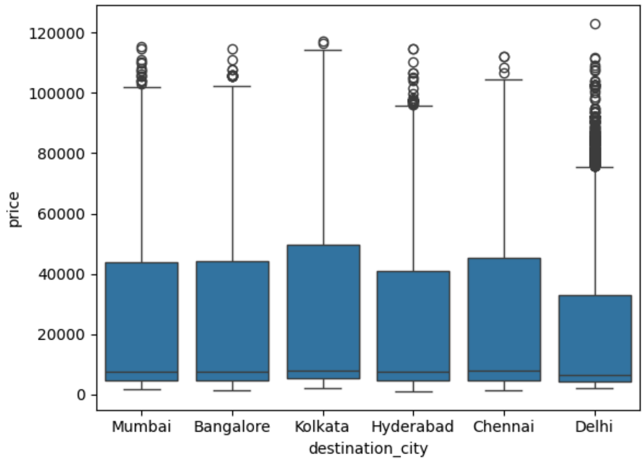


Rose A
6 May 2024

In the given plot below, it is observed that the price range vary for departure_time. This indicates the categories can vary with price hence feature can be used for prediction

```
sns.boxplot(x="destination_city", y="price", data=dt )
```

```
<Axes: xlabel='destination_city', ylabel='price'>
```



Rose A
6 May 2024
(edited 6 May 2024)

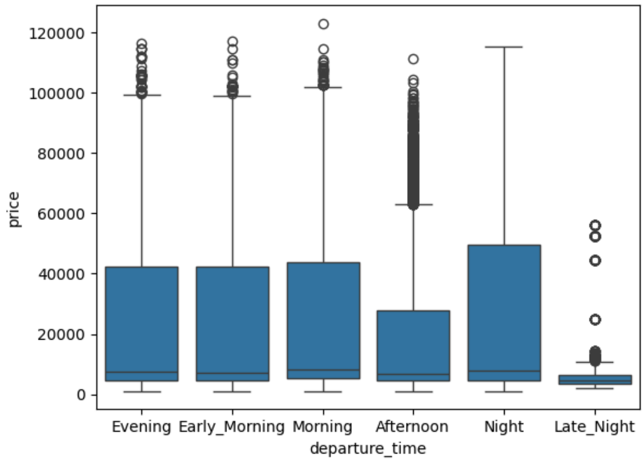
In the given plot below, it is observed that the price range vary for stops. This indicates the categories can vary with price hence feature can be used for prediction

Rose A
6 May 2024

have wider price ranges between one another. This feature is also crucial for price prediction.

```
sns.boxplot(x="departure_time", y="price", data=dt )
```

```
<Axes: xlabel='departure_time',
ylabel='price'>
```



Rose A
6 May 2024
(edited 6 May 2024)

vary

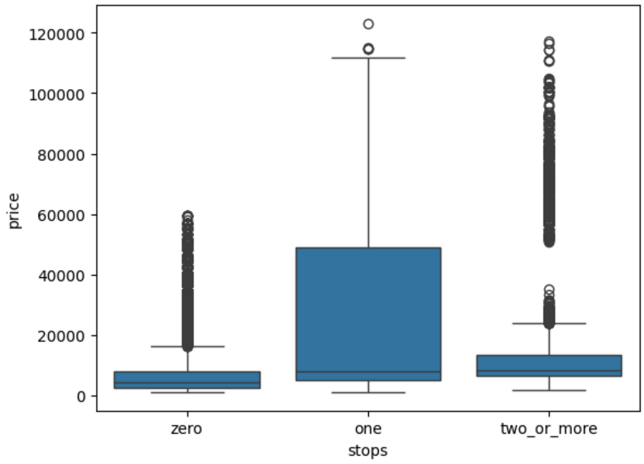
Rose A
6 May 2024

have wider price ranges between one another. This feature is also crucial for price prediction.

Start coding or generate with AI.

```
sns.boxplot(x="stops", y="price", data=dt )
```

```
<Axes: xlabel='stops', ylabel='price'>
```

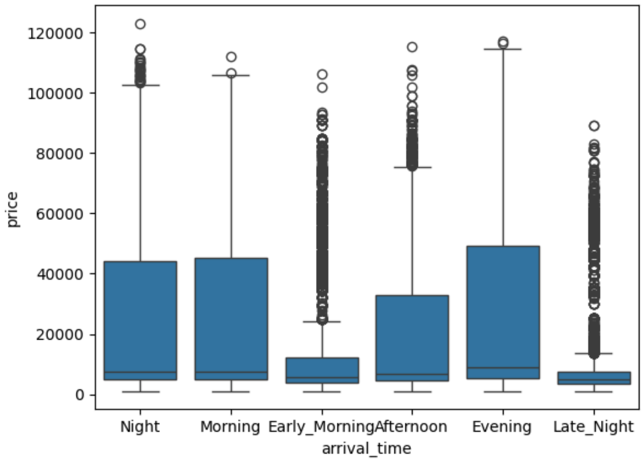


Rose A
6 May 2024

have wider price ranges between one another. This feature is also crucial for price prediction.

```
sns.boxplot(x="arrival_time", y="price", data=dt )
```

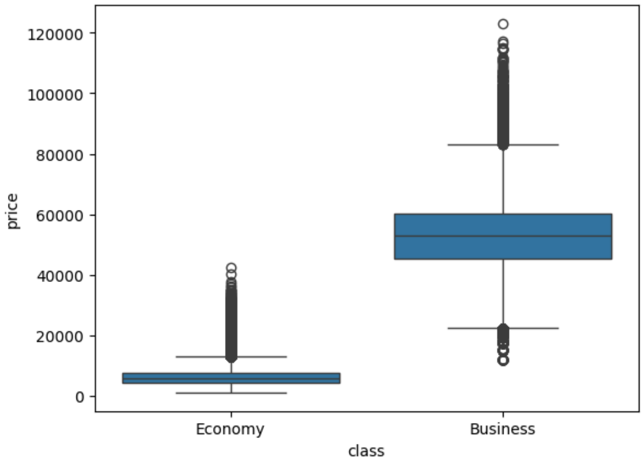
```
<Axes: xlabel='arrival_time',
ylabel='price'>
```



Start coding or generate with AI.

```
sns.boxplot(x="class", y="price", data=dt )
```

```
<Axes: xlabel='class', ylabel='price'>
```




```
dt.drop(['NO','flight',], axis = 1, inplace
```

```
dt.shape
```

```
(300153, 10)
```

```
#Data Transformation
```

```
from sklearn.preprocessing import LabelEncc

labelencoder = LabelEncoder()
dt.days_left = labelencoder.fit_transform(c
dt.duration=labelencoder.fit_transform(dt.c
dt.airline=labelencoder.fit_transform(dt.ai
dt.source_city=labelencoder.fit_transform(c
dt.departure_time=labelencoder.fit_transfor
dt.stops=labelencoder.fit_transform(dt.stop
dt.arrival_time=labelencoder.fit_transform(
dt.destination_city=labelencoder.fit_transf
```

```
dt.columns = dt.columns.str.replace('class'
#change the name of the feature because it'
```

```
dt.columns = dt.columns.str.replace('class_
```

```
dt.columns #check the columns name
```

```
Index(['airline', 'source_city',
'departure_time', 'stops',
'arrival_time',
       'destination_city',
'class_Type', 'duration', 'days_left',
'price'],
      dtype='object')
```

```
dt.class_Type=labelencoder.fit_transform(dt
```

```
dt.head(10)
```

	airline	source_city	departure_time
0	4	2	2
1	4	2	1
2	0	2	1
3	5	2	4
4	5	2	4
5	5	2	4
6	5	2	4
7	5	2	0
8	2	2	1
9	2	2	0

```
# Calculate the z-score with scipy
import scipy.stats as stats
dt = stats.zscore(dt)
```

```
dt # check the result of z socre
```

	airline	source_city	departure
0	0.488270	-0.329721	-0.2
1	0.488270	-0.329721	-0.8
2	-1.693633	-0.329721	-0.8
3	1.033746	-0.329721	0.9
4	1.033746	-0.329721	0.9
...	...	...	
300148	1.033746	-0.900576	0.9
300149	1.033746	-0.900576	-1.3
300150	1.033746	-0.900576	-0.8
300151	1.033746	-0.900576	-0.8
300152	1.033746	-0.900576	0.9

300153 rows × 10 columns

Rose A
6 May 2024

Based on the MAE, it is concluded that the Random Forest is the best regression model for predicting the car price based on the 10 predictor variables

```
x_train=dt.iloc[:,0:8]
y_train=dt.iloc[:,9]
```

```
x_train.head()#check the train head feature
```

	airline	source_city	departure_time
0	0.488270	-0.329721	-0.237897
1	0.488270	-0.329721	-0.807934
2	-1.693633	-0.329721	-0.807934
3	1.033746	-0.329721	0.902176
4	1.033746	-0.329721	0.902176

```
y_train.head()# check the traget variable
```

```
0   -0.658068
1   -0.658068
2   -0.657936
3   -0.657980
4   -0.657980
Name: price, dtype: float64
```

```
#importing train_test_split from sklearn
from sklearn.model_selection import train_t
```

```
#Fit Model
#Multiple Linear Regression
#Calling multiple linear regression model a
from sklearn.linear_model import LinearRegr

model = LinearRegression()
model_mlr = model.fit(X_train,Y_train)
```

Double-click (or enter) to edit

```
Y_pred_MLR = model_mlr.predict(X_test)  #Ma
```

```
#Calculating the Mean Square Error for MLR
mse_MLR = mean_squared_error(Y_test, Y_prec
print('The mean square error for Multiple L
```

```
The mean square error for Multiple Line
```

```
#Calculating the Mean Absolute Error for ML
mae_MLR= mean_absolute_error(Y_test, Y_prec
print('The mean absolute error for Multiple
```

```
The mean absolute error for Multiple Li
```

```
#Random Forest Regressor (checking other Mc
#calling the random forest model and fittir
rfModel = RandomForestRegressor()
model_rf = rfModel.fit(X_train,Y_train)
```

```
Y_pred_RF = model_rf.predict(X_test)
```

```
#Random Forest Evaluation
#Calculating the Mean Square Error for Ranc
mse_RF = mean_squared_error(Y_test, Y_pred_
print('The mean square error of price and p
```

```
The mean square error of price and pred
```