

flight-price-predictions

January 30, 2024

Day97-98 Flight Price Prediction using ML Model By: Loga Aswin

Importing Libraries

```
[105]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Dataset

```
[106]: df = pd.read_excel("/content/Data_Train.xlsx")
```

Data Pre-processing

```
[107]: df.head()
```

```
[107]:
```

	Airline	Date_of_Journey	Source	Destination	Route	\
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	

	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	05:50	13:15	7h 25m	2 stops	No info	7662
2	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	18:05	23:30	5h 25m	1 stop	No info	6218
4	16:50	21:35	4h 45m	1 stop	No info	13302

```
[108]: df.drop('Date_of_Journey',axis=1,inplace=True)
```

Checking Null Values

```
[109]: df.isnull().sum()
```

```
[109]: Airline      0
Source        0
Destination   0
```

```

Route          1
Dep_Time       0
Arrival_Time   0
Duration       0
Total_Stops    1
Additional_Info 0
Price          0
dtype: int64

```

```
[110]: df.dropna(inplace=True)
```

```
[111]: df.columns
```

```
[111]: Index(['Airline', 'Source', 'Destination', 'Route', 'Dep_Time', 'Arrival_Time',
            'Duration', 'Total_Stops', 'Additional_Info', 'Price'],
            dtype='object')
```

```
[112]: df.isnull().any()
```

```
[112]: Airline      False
Source      False
Destination  False
Route       False
Dep_Time    False
Arrival_Time False
Duration    False
Total_Stops False
Additional_Info False
Price       False
dtype: bool

```

Extracting hours and mins from departure time

```
[113]: df[['Dep_Hour', 'Dep_Min']] = df['Dep_Time'].str.split(':', expand=True)
```

```
[63]: df.head()
```

```
[63]:
```

	Airline	Source	Destination	Route	Dep_Time	\
0	IndiGo	Banglore	New Delhi	BLR → DEL	22:20	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	18:05	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	16:50	

	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Dep_Hour	Dep_Min
0	01:10 22 Mar	2h 50m	non-stop	No info	3897	22	20
1	13:15	7h 25m	2 stops	No info	7662	05	50

2	04:25	10 Jun	19h	2 stops	No info	13882	09	25
3		23:30	5h 25m	1 stop	No info	6218	18	05
4		21:35	4h 45m	1 stop	No info	13302	16	50

```
[64]: df.drop('Dep_Time',axis=1,inplace=True)
df.head()
```

```
[64]:      Airline  Source Destination      Route  Arrival_Time \
0      IndiGo  Bangalore  New Delhi      BLR → DEL  01:10 22 Mar
1      Air India  Kolkata  Bangalore  CCU → IXR → BBI → BLR      13:15
2      Jet Airways  Delhi  Cochin  DEL → LKO → BOM → COK  04:25 10 Jun
3      IndiGo  Kolkata  Bangalore  CCU → NAG → BLR      23:30
4      IndiGo  Bangalore  New Delhi  BLR → NAG → DEL      21:35
```

	Duration	Total_Stops	Additional_Info	Price	Dep_Hour	Dep_Min
0	2h 50m	non-stop	No info	3897	22	20
1	7h 25m	2 stops	No info	7662	05	50
2	19h	2 stops	No info	13882	09	25
3	5h 25m	1 stop	No info	6218	18	05
4	4h 45m	1 stop	No info	13302	16	50

Extracting arrival hour and mins from arrival time

```
[65]: df['Arrival_Hour']=pd.to_datetime(df['Arrival_Time']).dt.hour
df['Arrival_min']=pd.to_datetime(df['Arrival_Time']).dt.minute
```

```
[66]: df.drop('Arrival_Time',axis=1,inplace=True)
```

```
[67]: duration=list(df['Duration'])
```

Making separate column for hour and min and extracting it from duration

```
[68]: duration_hours = [int(x.split('h')[0].strip()) if 'h' in x else 0 for x in
    ↪df['Duration']]
duration_mins = [int(x.split('m')[0].split()[-1].strip()) if 'm' in x else 0
    ↪for x in df['Duration']]
```

```
[69]: df['Duration_hours']=duration_hours
df['Duration_mins']=duration_mins
```

```
[70]: df.head()
```

```
[70]:      Airline  Source Destination      Route  Duration \
0      IndiGo  Bangalore  New Delhi      BLR → DEL  2h 50m
1      Air India  Kolkata  Bangalore  CCU → IXR → BBI → BLR  7h 25m
2      Jet Airways  Delhi  Cochin  DEL → LKO → BOM → COK  19h
3      IndiGo  Kolkata  Bangalore  CCU → NAG → BLR  5h 25m
```

4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m
---	--------	----------	-----------	-----------------	--------

	Total_Stops	Additional_Info	Price	Dep_Hour	Dep_Min	Arrival_Hour	\
0	non-stop	No info	3897	22	20	1	
1	2 stops	No info	7662	05	50	13	
2	2 stops	No info	13882	09	25	4	
3	1 stop	No info	6218	18	05	23	
4	1 stop	No info	13302	16	50	21	

	Arrival_min	Duration_hours	Duration_mins
0	10	2	50
1	15	7	25
2	25	19	0
3	30	5	25
4	35	4	45

```
[71]: df.drop('Duration',axis=1,inplace=True)
df.head()
```

	Airline	Source	Destination	Route	Total_Stops	\
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	

	Additional_Info	Price	Dep_Hour	Dep_Min	Arrival_Hour	Arrival_min	\
0	No info	3897	22	20	1	10	
1	No info	7662	05	50	13	15	
2	No info	13882	09	25	4	25	
3	No info	6218	18	05	23	30	
4	No info	13302	16	50	21	35	

	Duration_hours	Duration_mins
0	2	50
1	7	25
2	19	0
3	5	25
4	4	45

```
[72]: df['Airline'].value_counts()
```

Jet Airways	3849
IndiGo	2053
Air India	1751
Multiple carriers	1196
SpiceJet	818

```
Vistara          479
Air Asia         319
GoAir            194
Multiple carriers Premium economy    13
Jet Airways Business                   6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

handling categorical data

```
[73]: Airline=df[['Airline']]
      Airline=pd.get_dummies(Airline,drop_first=True)
```

performing one hot encoding on Source

```
[74]: df['Source'].value_counts()
```

```
[74]: Delhi          4536
      Kolkata        2871
      Bangalore      2197
      Mumbai         697
      Chennai        381
      Name: Source, dtype: int64
```

```
[75]: Source=df[['Source']]
      Source=pd.get_dummies(Source,drop_first=True)
      Source.head()
```

```
[75]:   Source_Chennai  Source_Delhi  Source_Kolkata  Source_Mumbai
0                0              0                0                0
1                0              0                1                0
2                0              1                0                0
3                0              0                1                0
4                0              0                0                0
```

```
[76]: Destination=df[['Destination']]
      Destination=pd.get_dummies(Destination, drop_first=True)
```

```
[77]: df.drop(['Route', 'Additional_Info'],axis=1,inplace=True)
```

```
[78]: df['Total_Stops'].value_counts()
```

```
[78]: 1 stop          5625
      non-stop       3491
      2 stops        1520
      3 stops         45
      4 stops         1
```

Name: Total_Stops, dtype: int64

performing label encoding on ordinal stops data

```
[79]: df.replace({'non-stop':0, '1 stop':1, '2 stops':2, '3 stops':3, '4 stops':  
↪4},inplace=True)
```

```
[80]: df.head()
```

```
[80]:
```

	Airline	Source	Destination	Total_Stops	Price	Dep_Hour	Dep_Min	\
0	IndiGo	Banglore	New Delhi	0	3897	22	20	
1	Air India	Kolkata	Banglore	2	7662	05	50	
2	Jet Airways	Delhi	Cochin	2	13882	09	25	
3	IndiGo	Kolkata	Banglore	1	6218	18	05	
4	IndiGo	Banglore	New Delhi	1	13302	16	50	

	Arrival_Hour	Arrival_min	Duration_hours	Duration_mins
0	1	10	2	50
1	13	15	7	25
2	4	25	19	0
3	23	30	5	25
4	21	35	4	45

```
[81]: df_final=pd.concat([df,Airline,Source,Destination],axis=1)  
df_final.head()
```

```
[81]:
```

	Airline	Source	Destination	Total_Stops	Price	Dep_Hour	Dep_Min	\
0	IndiGo	Banglore	New Delhi	0	3897	22	20	
1	Air India	Kolkata	Banglore	2	7662	05	50	
2	Jet Airways	Delhi	Cochin	2	13882	09	25	
3	IndiGo	Kolkata	Banglore	1	6218	18	05	
4	IndiGo	Banglore	New Delhi	1	13302	16	50	

	Arrival_Hour	Arrival_min	Duration_hours	...	\
0	1	10	2	...	
1	13	15	7	...	
2	4	25	19	...	
3	23	30	5	...	
4	21	35	4	...	

	Airline_Vistara	Premium economy	Source_Chennai	Source_Delhi	\
0		0	0	0	
1		0	0	0	
2		0	0	1	
3		0	0	0	
4		0	0	0	

	Source_Kolkata	Source_Mumbai	Destination_Cochin	Destination_Delhi	\
0	0	0	0	0	
1	1	0	0	0	
2	0	0	1	0	
3	1	0	0	0	
4	0	0	0	0	

	Destination_Hyderabad	Destination_Kolkata	Destination_New Delhi
0	0	0	1
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	1

[5 rows x 31 columns]

```
[82]: df_final.shape
```

```
[82]: (10682, 31)
```

```
[83]: df_final.drop(['Airline', 'Source', 'Destination'], axis=1, inplace=True)
df_final.head()
```

```
[83]:
```

	Total_Stops	Price	Dep_Hour	Dep_Min	Arrival_Hour	Arrival_min	\
0	0	3897	22	20	1	10	
1	2	7662	05	50	13	15	
2	2	13882	09	25	4	25	
3	1	6218	18	05	23	30	
4	1	13302	16	50	21	35	

	Duration_hours	Duration_mins	Airline_Air India	Airline_GoAir	...	\
0	2	50	0	0	...	
1	7	25	1	0	...	
2	19	0	0	0	...	
3	5	25	0	0	...	
4	4	45	0	0	...	

	Airline_Vistara Premium economy	Source_Chennai	Source_Delhi	\
0	0	0	0	
1	0	0	0	
2	0	0	1	
3	0	0	0	
4	0	0	0	

	Source_Kolkata	Source_Mumbai	Destination_Cochin	Destination_Delhi	\
0	0	0	0	0	
1	1	0	0	0	

2	0	0	1	0
3	1	0	0	0
4	0	0	0	0

	Destination_Hyderabad	Destination_Kolkata	Destination_New Delhi
0	0	0	1
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	1

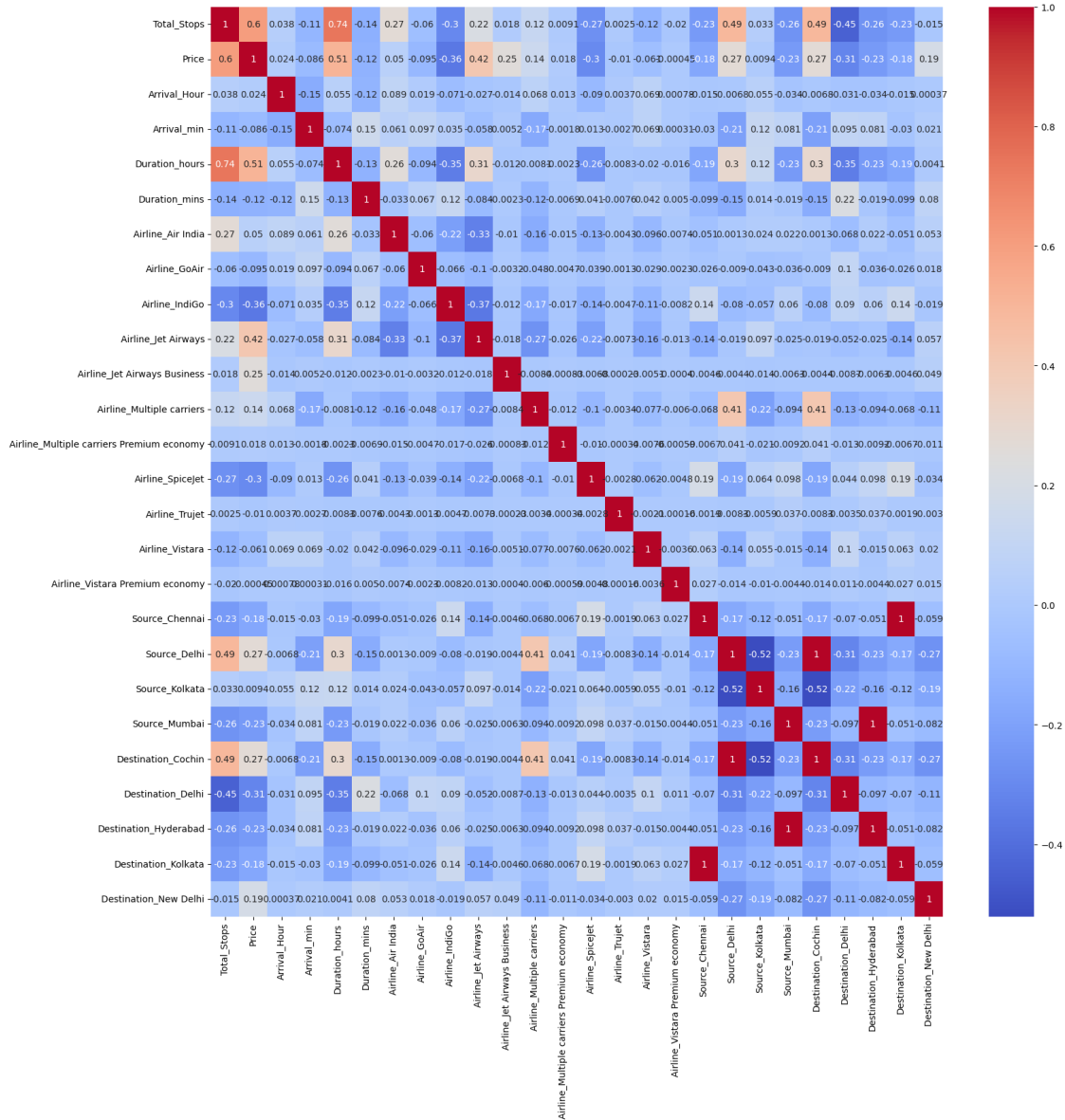
[5 rows x 28 columns]

```
[84]: X=df_final.drop('Price',axis=1)
      y=df_final['Price']
```

```
[85]: plt.figure(figsize=(18,18))
      sns.heatmap(df_final.corr(),annot=True,cmap='coolwarm')
      plt.show()
```

<ipython-input-85-b09baae2a6a6>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

```
sns.heatmap(df_final.corr(),annot=True,cmap='coolwarm')
```

Feature Selection using GradientBoostingRegressor

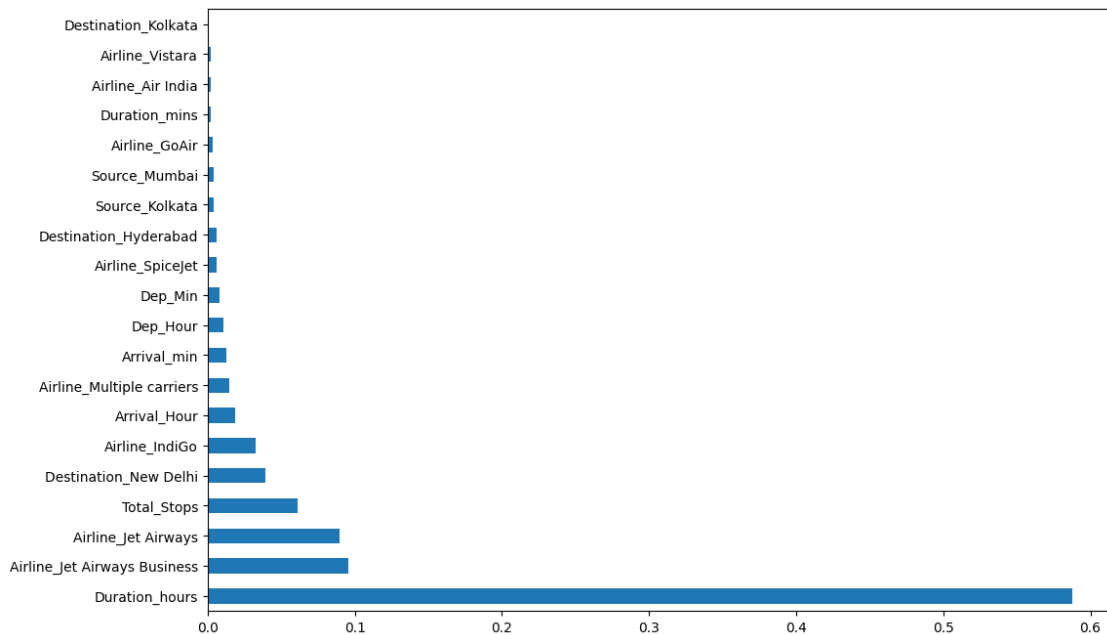
```
[86]: from sklearn.ensemble import GradientBoostingRegressor
```

```
model = GradientBoostingRegressor()
model.fit(X, y)
print(model.feature_importances_)
```

```
[6.12343759e-02 1.03241578e-02 7.67889092e-03 1.87781678e-02
1.24947486e-02 5.87490478e-01 1.84071819e-03 1.82307121e-03
3.49512915e-03 3.23417076e-02 8.97082873e-02 9.57407051e-02
1.46863950e-02 2.32679713e-04 6.18538948e-03 0.00000000e+00]
```

```
1.70450723e-03 6.17524302e-05 2.17585137e-05 2.57634518e-04
4.08857903e-03 3.76077732e-03 1.79614368e-04 8.62803712e-05
6.09050229e-03 8.21077000e-04 3.88726152e-02]
```

```
[87]: plt.figure(figsize=(12,8))
model=pd.Series(model.feature_importances_,index=X.columns)
model.nlargest(20).plot(kind='barh')
plt.show()
```



Building Model using Random Forest

```
[88]: from sklearn.model_selection import train_test_split
```

```
[89]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪2,random_state=42)
```

```
[90]: from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor()
model.fit(X_train,y_train)
```

```
[90]: RandomForestRegressor()
```

```
[91]: y_pred=model.predict(X_test)
```

```
[92]: model.score(X_train,y_train)
```

```
[92]: 0.7888808655761974
```

```
[93]: model.score(X_test,y_test)
```

```
[93]: 0.5711694875328623
```

```
[94]: n_estimators=[int(x) for x in np.linspace(start=100, stop=1200, num=12)]
max_features=['auto','sqrt']
max_depth=[int(x) for x in np.linspace(5,30,num=6)]
min_samples_split=[2,5,10,15,100]
min_samples_leaf=[1,2,5,10]
```

```
[95]: random_grid={
    'n_estimators':n_estimators,
    'max_features':max_features,
    'max_depth':max_depth,
    'min_samples_split':min_samples_split,
    'min_samples_leaf':min_samples_leaf
}
```

```
[96]: from sklearn.model_selection import RandomizedSearchCV
```

```
[97]: model2=RandomizedSearchCV(estimator=model, param_distributions=random_grid,
    ↪scoring='neg_mean_squared_error',n_iter=10, cv=4,random_state=42,n_jobs=1)
```

```
[98]: model2.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
```

```
    warn(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
```

```
    warn(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
```

```
    warn(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
```

and ExtraTreesRegressors.

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors
```

and ExtraTreesRegressors.

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:  
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or  
remove this parameter as it is also the default value for RandomForestRegressors  
and ExtraTreesRegressors.
```

```
warn(  

```

```
[98]: RandomizedSearchCV(cv=4, estimator=RandomForestRegressor(), n_jobs=1,  
    param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],  
        'max_features': ['auto', 'sqrt'],  
        'min_samples_leaf': [1, 2, 5, 10],  
        'min_samples_split': [2, 5, 10, 15,  
                                100],  
        'n_estimators': [100, 200, 300, 400,  
                           500, 600, 700, 800,  
                           900, 1000, 1100,  
                           1200]},  
    random_state=42, scoring='neg_mean_squared_error')
```

```
[99]: model2.best_params_
```

```
[99]: {'n_estimators': 700,  
      'min_samples_split': 15,  
      'min_samples_leaf': 1,  
      'max_features': 'auto',  
      'max_depth': 20}
```

```
[100]: prediction=model.predict(X_test)
```

```
[101]: from sklearn import metrics  
      print(metrics.r2_score(y_test,prediction))
```

0.5711694875328623

Predicting flight ticket costs can pose a challenge; what we observe today may not align with tomorrow's prices, creating an unpredictable scenario. Travelers frequently says that the unpredictable nature of flight ticket pricing.

This analysis tackles the machine learning task of forecasting flight ticket prices using various independent features.