

Problem Statement:-

Claculating price based on no.of stops

```
In [1]: 1 import numpy as np
        2 import pandas as pd
        3 import matplotlib.pyplot as plt
        4 import seaborn as sns
        5 from sklearn.model_selection import train_test_split
        6 from sklearn.linear_model import LinearRegression
```

```
In [2]: 1 df=pd.read_excel(r"C:\Users\HP\Downloads\Data_Train.xlsx")
        2 df
```

Out[2]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 11 columns

```
In [3]: 1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Airline                10683 non-null  object
1   Date_of_Journey        10683 non-null  object
2   Source                 10683 non-null  object
3   Destination            10683 non-null  object
4   Route                  10682 non-null  object
5   Dep_Time               10683 non-null  object
6   Arrival_Time           10683 non-null  object
7   Duration               10683 non-null  object
8   Total_Stops            10682 non-null  object
9   Additional_Info        10683 non-null  object
10  Price                  10683 non-null  int64
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

```
In [4]: 1 df.fillna(method='ffill',inplace=True)
```

In [5]:

1

df.isnull().sum()

Out[5]:

Airline	0
Date_of_Journey	0
Source	0
Destination	0
Route	0
Dep_Time	0
Arrival_Time	0
Duration	0
Total_Stops	0
Additional_Info	0
Price	0

dtype: int64

In [6]:

1

df['Total\_Stops'].value\_counts()

Out[6]:

Total_Stops	
1 stop	5625
non-stop	3492
2 stops	1520
3 stops	45
4 stops	1

Name: count, dtype: int64

In [7]:

1

convert = {'Total\_Stops':{'1 stop':1,'2 stops':2,'3 stops':3,'4 stops':4,'non-stop':5}}

2

df=df.replace(convert)

3

df

Out[7]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	5	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30m	5	No info	4107
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35m	5	No info	4145
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h	5	No info	7229
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40m	5	No info	12648
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20m	2	No info	11753

10683 rows × 11 columns

In [8]:

1

df.columns

Out[8]:

Index(['Airline', 'Date\_of\_Journey', 'Source', 'Destination', 'Route', 'Dep\_Time', 'Arrival\_Time', 'Duration', 'Total\_Stops', 'Additional\_Info', 'Price'], dtype='object')

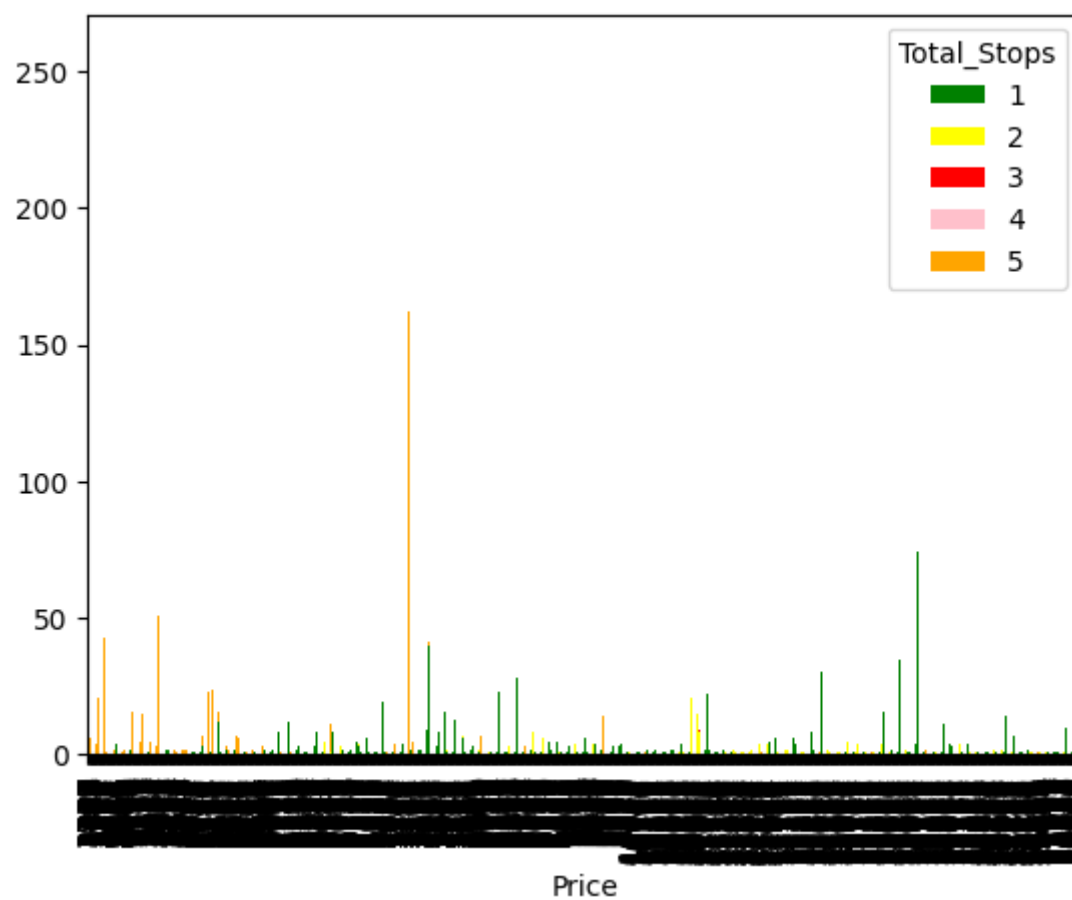
```
In [9]: 1 c=pd.crosstab(df['Price'],df['Total_Stops'])
        2 print(c)
```

```
Total_Stops  1  2  3  4  5
Price
1759          0  0  0  0  4
1840          0  0  0  0  1
1965          0  0  0  0 36
2017          0  0  0  0 35
2050          0  0  0  0 10
...
52285         ..  ..  ..  ..  ..
52285         0  1  0  0  0
54826         3  0  0  0  0
57209         1  0  0  0  0
62427         1  0  0  0  0
79512         1  0  0  0  0
```

[1870 rows x 5 columns]

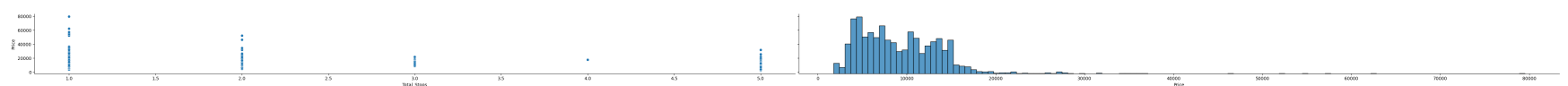
```
In [10]: 1 c.plot(kind='bar', stacked=True, color=['green','yellow','red','pink','orange'],grid=False)
```

Out[10]: <Axes: xlabel='Price'>



```
In [11]: 1 sns.pairplot(df,x_vars=['Total_Stops','Price'],y_vars=['Price'],aspect=10)
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x227695657b0>



```
In [12]: 1 x=df[['Total_Stops','Price']]
        2 y=df[['Price']]
```

```
In [13]: 1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=10)
        2 x_train.shape,x_test.shape
```

Out[13]: ((7478, 2), (3205, 2))

```
In [14]: 1 lr=LinearRegression()
        2 lr.fit(x_train,y_train)
        3 print('Linear Regression')
        4 print(lr.score(x_test,y_test))
        5 print(lr.score(x_train,y_train))
```

Linear Regression

1.0

1.0

```
In [15]: 1 from sklearn.linear_model import Ridge,Lasso
2 lasso = Lasso(alpha=10)
3 lasso.fit(x_train,y_train)
4 print('Lasso regularisation model score')
5 print(lasso.score(x_train,y_train))
6 print(lasso.score(x_test,y_test))
```

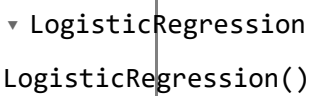
Lasso regularisation model score  
0.9999999999997754  
0.9999999999997753

```
In [16]: 1 from sklearn.linear_model import Ridge,Lasso
2 ridge = Lasso(alpha=10)
3 ridge.fit(x_train,y_train)
4 print('Ridge regularisation model score')
5 print(ridge.score(x_train,y_train))
6 print(ridge.score(x_test,y_test))
```

Ridge regularisation model score  
0.9999999999997754  
0.9999999999997753

```
In [17]: 1 import warnings
2 warnings.simplefilter(action='ignore')
```

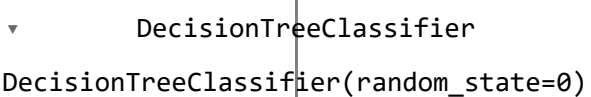
```
In [18]: 1 from sklearn.linear_model import LogisticRegression
2 lig=LogisticRegression()
3 lig.fit(x_train,y_train)
```

Out[18]: 
  
LogisticRegression()  
LogisticRegression()

```
In [19]: 1 print('Logistic Regression')
2 print(lig.score(x_train,y_train))
3 print(lig.score(x_test,y_test))
```

Logistic Regression  
0.024204332709280556  
0.024024960998439936

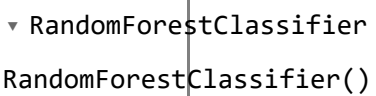
```
In [20]: 1 from sklearn.tree import DecisionTreeClassifier
2 dt=DecisionTreeClassifier(random_state=0)
3 dt.fit(x_train,y_train)
```

Out[20]: 
  
DecisionTreeClassifier  
DecisionTreeClassifier(random\_state=0)

```
In [21]: 1 print(dt.score(x_train,y_train))
2 print(dt.score(x_test,y_test))
```

1.0  
0.8895475819032761

```
In [22]: 1 from sklearn.ensemble import RandomForestClassifier
2 rfc=RandomForestClassifier()
3 rfc.fit(x_train,y_train)
```

Out[22]: 
  
RandomForestClassifier  
RandomForestClassifier()

```
In [23]: 1 print(rfc.score(x_train,y_train))
2 print(rfc.score(x_test,y_test))
```

1.0  
0.8854914196567862

```
In [24]: 1 rf=RandomForestClassifier()
```

```
In [32]: 1 params={'max_depth':[2,4,6,7,9], 'min_samples_leaf':[5,10,20,30,50,100], 'n_estimators':[10,20,30,40,100,200]}
```

```
In [33]: 1 from sklearn.model_selection import GridSearchCV
```

```
In [42]: 1 grid_search = GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
2 grid_search.fit(x_test,y_test)
```

Out[42]:

GridSearchCV

estimator: RandomForestClassifier

RandomForestClassifier

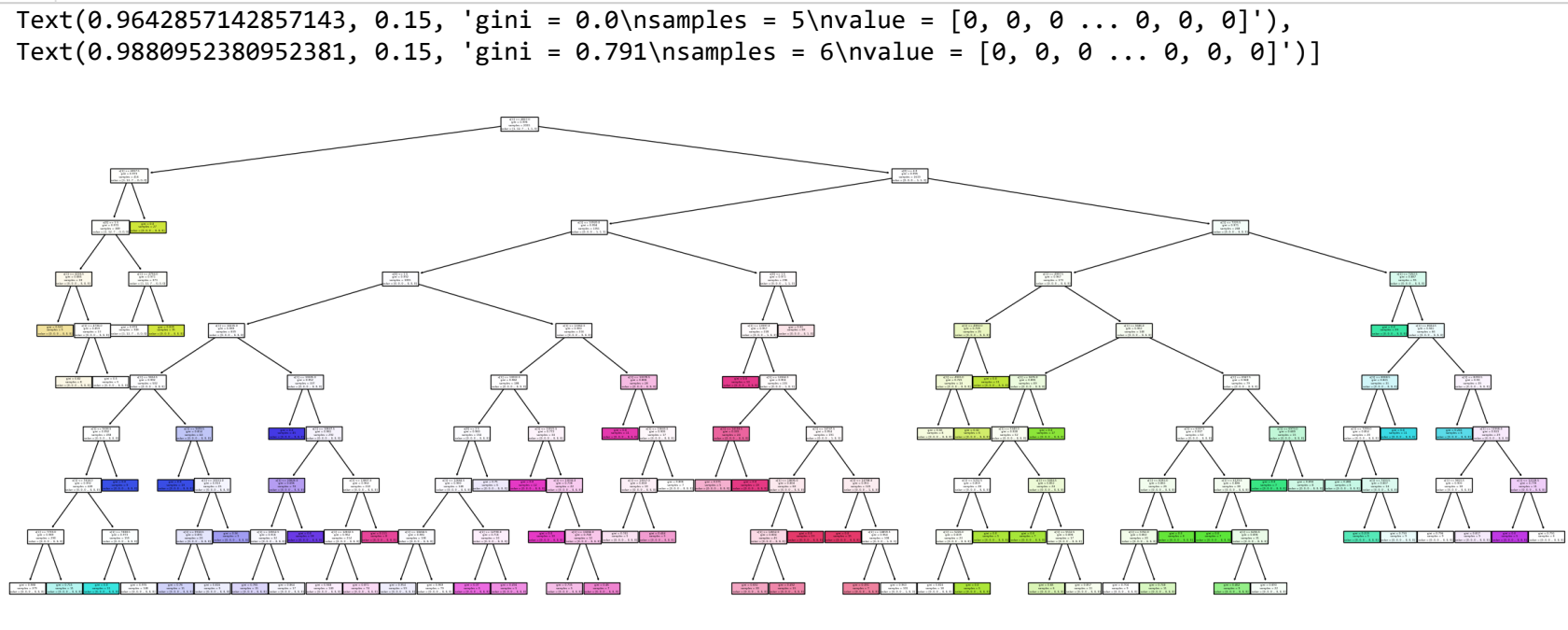
```
In [43]: 1 grid_search.best_score_
```

Out[43]: 0.5166948597472124

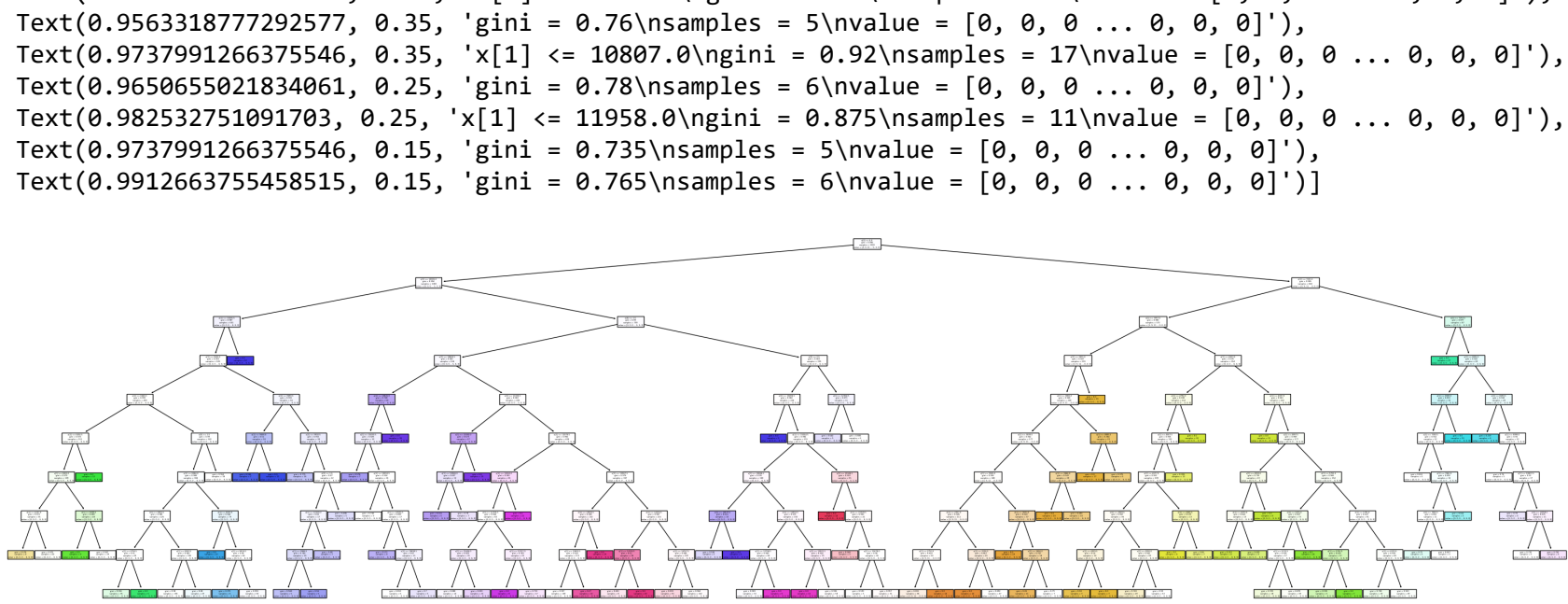
```
In [44]: 1 rf_best = grid_search.best_estimator_
2 print(rf_best)
```

RandomForestClassifier(max\_depth=9, min\_samples\_leaf=5)

```
In [45]: 1 from sklearn.tree import plot_tree
2 plt.figure(figsize=(80,10))
3 plot_tree(rf_best.estimators_[5],filled=True)
```



```
In [46]: 1 from sklearn.tree import plot_tree
2 plt.figure(figsize=(40,10))
3 plot_tree(rf_best.estimators_[7],filled =True)
```



```
In [47]: 1 rf_best.feature_importances_
```

Out[47]: array([0.03656111, 0.96343889])

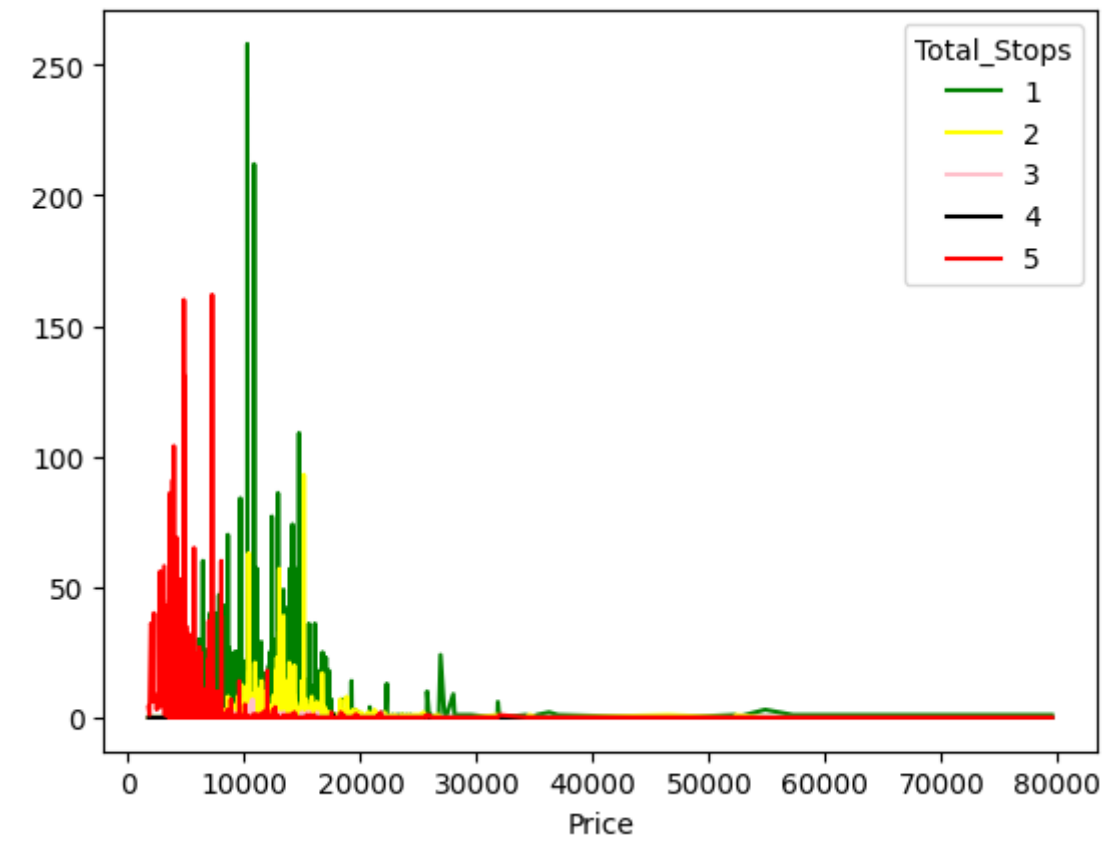
```
In [40]: 1 imp_df = pd.DataFrame({'Varname':x_test.columns,'Imp':rf_best.feature_importances_})
2 imp_df.sort_values(by='Imp',ascending = False)
```

Out[40]:

	Varname	Imp
1	Price	0.964642
0	Total_Stops	0.035358

```
In [41]: 1 c.plot(kind='line', stacked=False, color=['green','yellow','pink','black','red'],grid=False)
```

Out[41]: <Axes: xlabel='Price'>



Conclusion:-

If The airline having 1 stop or non-stop price is more compared to 2-stops and 3-stops

```
In [ ]: 1
```