PROBLEM STATEMENT: Which model is suitable for flight price prediction

In [1]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import Ridge,RidgeCV,Lasso
from sklearn.linear_model import ElasticNet
```

In [2]:
```
pip install openpyxl
```

```
Defaulting to user installation because normal site-packages is not writeabl
e
Requirement already satisfied: openpyxl in c:\users\hp\appdata\roaming\pytho
n\python310\site-packages (3.1.2)
Requirement already satisfied: et-xmlfile in c:\users\hp\appdata\roaming\pyt
hon\python310\site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
df=pd.read_excel(r"C:\Users\HP\Downloads\Data_Train.xlsx")
df
```

Out[3]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50n
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25n
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25n
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45n
...	...	...	...	...	...	...	...	..
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30n
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35n
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40n
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20n

10683 rows × 11 columns

In [4]:
```
df.head()
```

Out[4]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	To
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	

In [5]:
```
df.tail()
```

Out[5]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30n
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35n
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3l
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40n
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20n

In [6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

In [8]:
```
df.shape
```

Out[8]:
```
(10683, 11)
```

In [9]:
```
convert={"Total_Stops":{"1 stop":1,"2 stops":2,"non-stop":3,"3 stops":4,"4 sto
df=df.replace(convert)
df
```

Out[9]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duratio
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50n
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25n
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19l
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25n
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45n
...	...	...	...	...	...	...	...	..
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30n
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35n
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3l
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40n
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20n

10683 rows × 11 columns

In [10]:
```
features=df['Total_Stops']
target=df.columns[-1]
```

In [11]:
```
df=df[['Total_Stops','Price']]
df.columns=['TS','prc']
```

In [12]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11244\4116506308.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

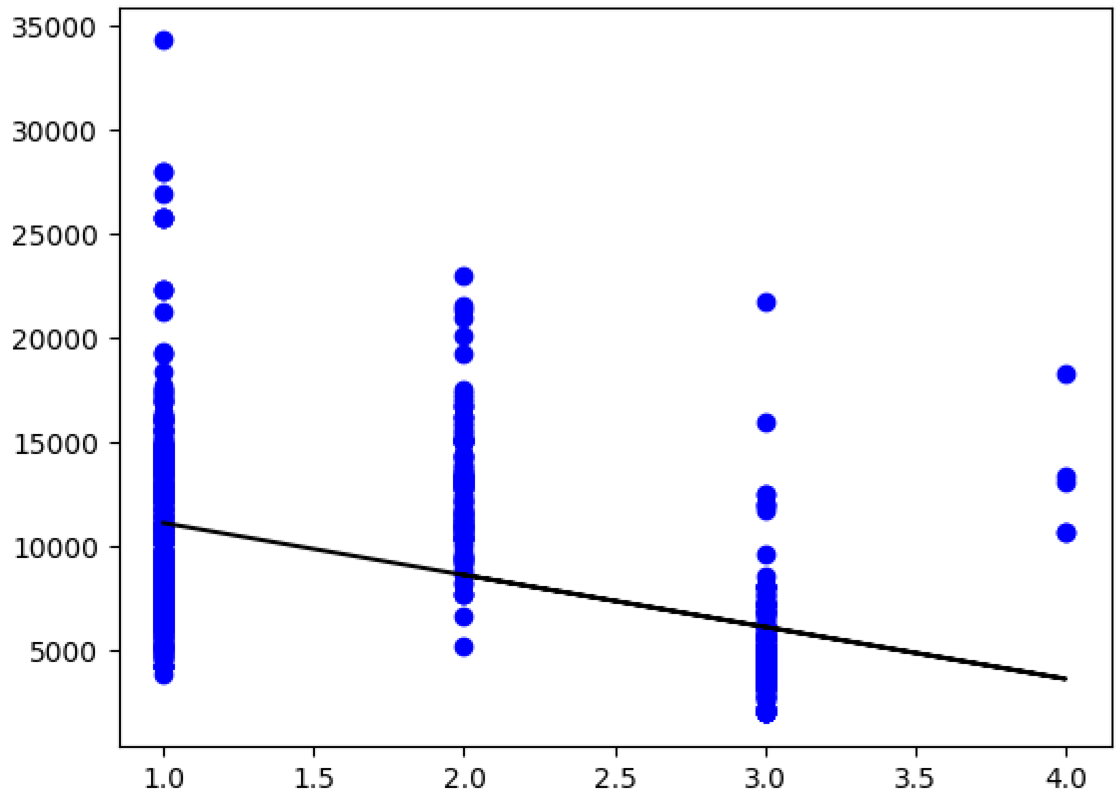
In [13]:
```
X = np.array(df['TS']).reshape(-1,1)
y = np.array(df['prc']).reshape(-1,1)
```

In [14]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [15]:
```
X_train,x_test,y_train,y_test = train_test_split(X,y,train_size=0.9)
regr = LinearRegression()
regr.fit(X_train,y_train)
print(regr.score(x_test, y_test))
```

```
0.25354171235463285
```

In [16]:
```
y_pred = regr.predict(x_test)
plt.scatter(x_test, y_test, color ='b')
plt.plot(x_test, y_pred, color ='k')
plt.show()
```



In [17]:
```
coeff_df=pd.DataFrame(regr.coef_)
coeff_df
```

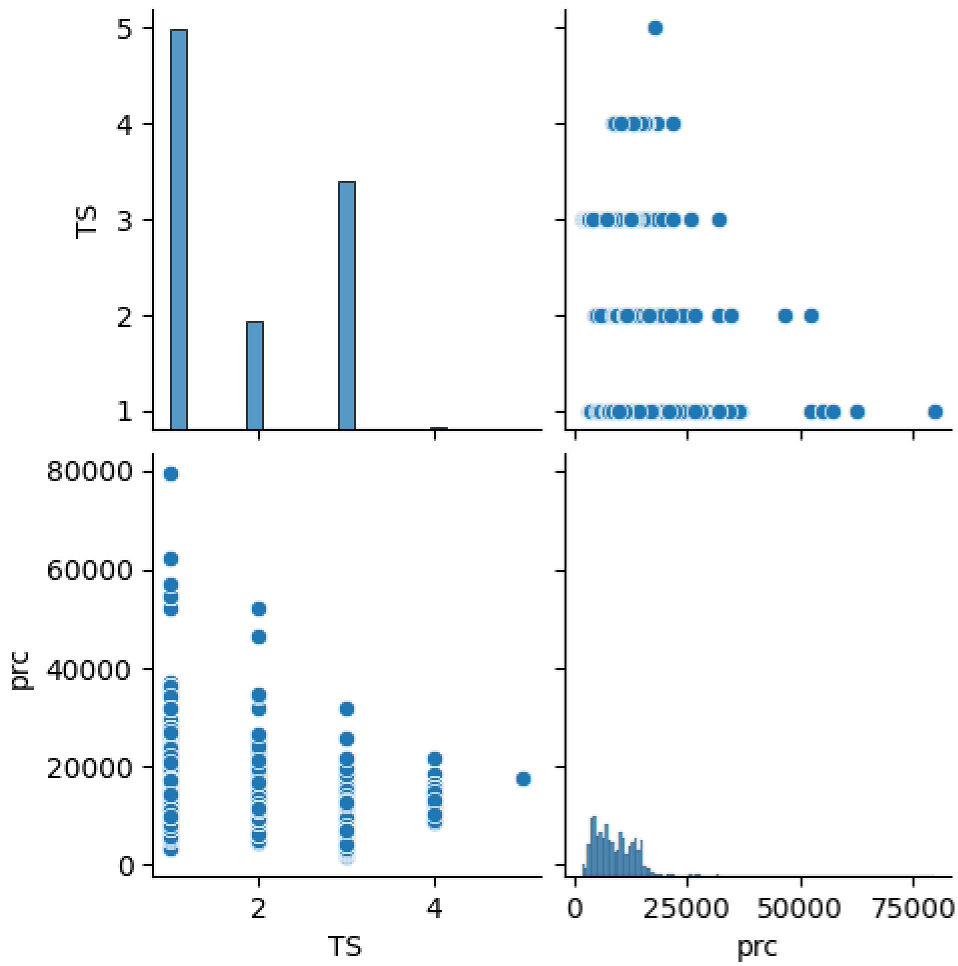
Out[17]:

	0
0	-2498.76426

Exploratory Data Set

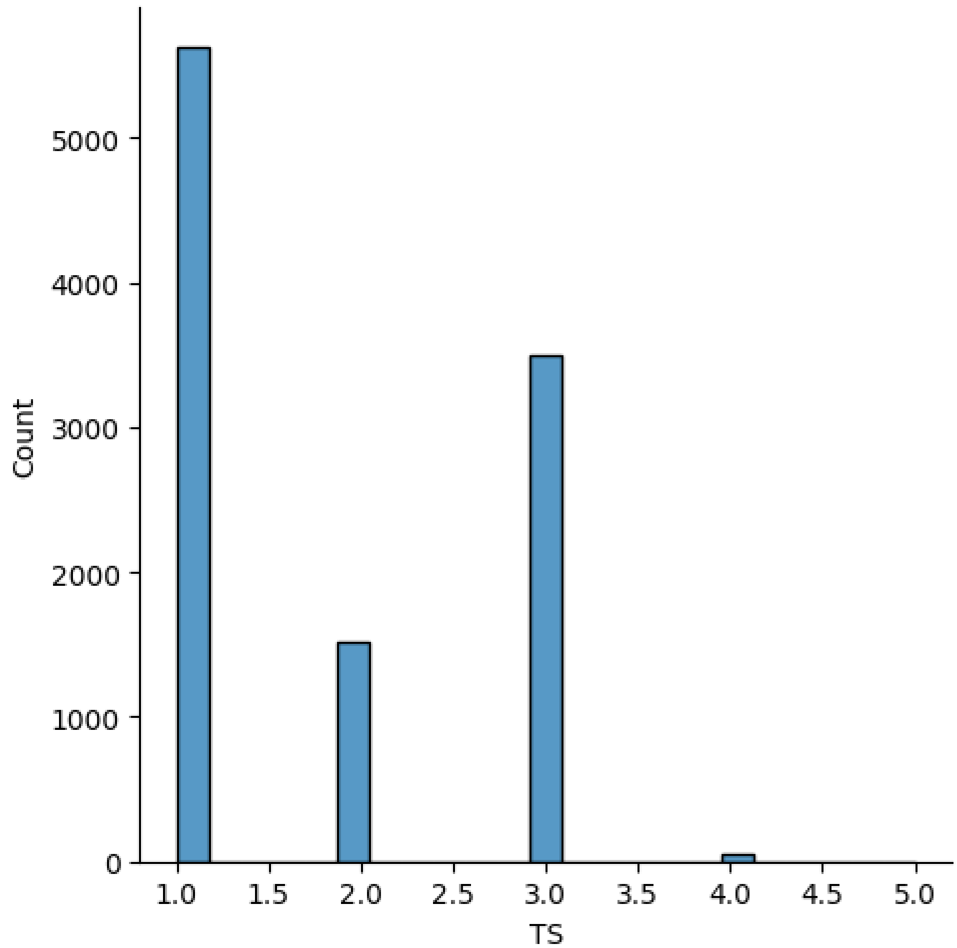
In [18]:
```
sns.pairplot(df)
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x1b12a0ab040>



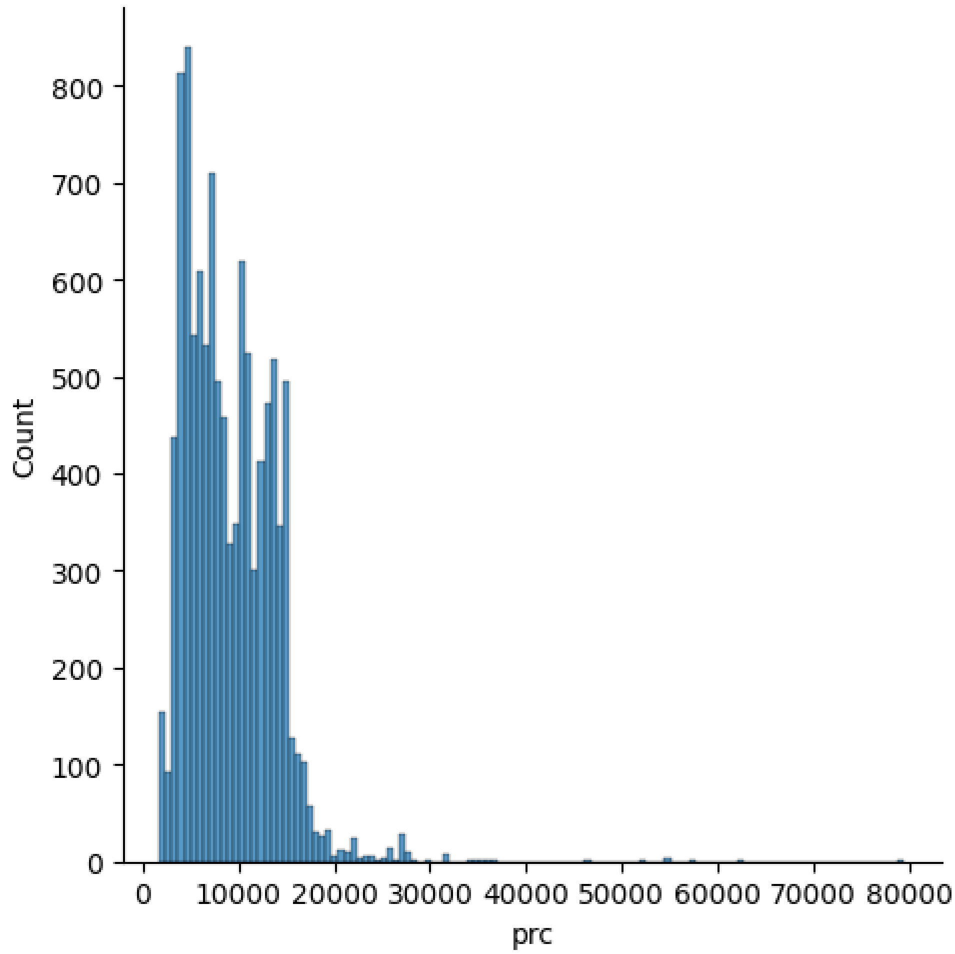
In [19]:
```
sns.displot(df['TS'])
```

Out[19]: <seaborn.axisgrid.FacetGrid at 0x1b12a125150>



In [24]:
```
sns.displot(df['prc'])
```

Out[24]: <seaborn.axisgrid.FacetGrid at 0x1b12a33ffa0>



Ridge

In [21]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train,y_train)
train_score_ridge = ridgeReg.score(X_train,y_train)
test_score_ridge = ridgeReg.score(x_test,y_test)
print('\nRidge model\n')
print('Train score for ridge model is {}'.format(train_score_ridge))
print('Test score for ridge model is {}'.format(test_score_ridge))
```

```

Ridge model

Train score for ridge model is 0.24385789070810748
Test score for ridge model is 0.2535436118510105
```

Lasso

In [22]:
```
lassoReg=Lasso(alpha=10)
lassoReg.fit(X_train,y_train)
train_score_lasso=lassoReg.score(X_train,y_train)
test_score_lasso=lassoReg.score(x_test,y_test)
print('\nLasso Model\n')
print('Train score for lasso model is {}'.format(train_score_lasso))
print('Test score for lasso model is {}'.format(test_score_lasso))
```

```

Lasso Model

Train score for lasso model is 0.2438526870271216
Test score for lasso model is 0.2535446638790251
```

ELASTIC NET

In [23]:
```
regr = ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
y_pred_elastic = regr.predict(X_train)
mean_squared_error = np.mean((y_pred_elastic-y_train)**2)
print('Mean squared error on test set',mean_squared_error)
regr.score(X_train,y_train)
```

```
[-1561.68685919]
[11912.22187033]
Mean squared error on test set 23462267.300860338
```

Out[23]:
```
0.20955401822999142
```

Conclusion:Elastic Net is the best model for this data set.

In [ ]:
```

```