Projekt Flugpreis vorhersagen

March 18, 2022

flugpreis Vorhersage

ein KI-Modell wurde entwickelt, um den Flugpreis auf verschiedenen Strecken vorherzusagen.

0.1 Daten verstehen und bearbeiten

[1]:
```
# notwendige packet importieren

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
pd.options.mode.chained_assignment = None  # default='warn'
```

Da die Daten in Form einer Excel-Datei vorliegen, müssen wir Pandas read_excel verwenden, um die Daten zu laden Nach dem Laden ist es wichtig, Nullwerte in der Spalten oder Zeilen zu überprüfen Wenn es vorhanden ist, kann Folgendes getan werden:—> a.Füllen von NaN-Werten mit Mittelwert, Median und Modus unter Verwendung der fillna()-Methode—> b.Wenn weniger fehlende Werte vorhanden sind, können wir diese ebenfalls löschen.

[2]:
```
## train daten einlesen

train_data = pd.read_excel('Data_Train.xlsx')
```

[3]:
```
# die erste 5 spalten anschauen
train_data.head()
```

[3]:

	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						

```
4    16:50         21:35   4h 45m      1 stop         No info   13302
```

0.2 Umgang mit fehlende Werte

0.3 Datenbereinigung, um die Daten für die Analyse vorzubereiten

[4]:
```
train_data.shape
```

[4]:
```
(10683, 11)
```

[5]:
```
## gucken ob irgendwo null werte gibt in der dataframe
train_data.isna().sum()
```

[5]:
```
Airline            0
Date_of_Journey    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
```

Da weniger fehlende Werte vorhanden sind, nab kann diese direkt löschen

[6]:
```
# null spalten entfernen
train_data.dropna(inplace=True)
```

[7]:
```
# data frame nullwerte noch mal anschauen
train_data.isna().sum()
```

[7]:
```
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
```

[8]:
```
# die daten type der spalte anschauen
train_data.dtypes
```

[8]:
```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Additional_Info    object
Price               int64
dtype: object
```

Aus der Beschreibung können wir ersehen, dass Date_of_Journey ein Objektdatentyp ist, Daher müssen wir diesen Datentyp zu Zeit umwandeln, um diese Spalte richtig für die Vorhersage später zu verwenden. Das Modell kann diese String-Werte nicht verstehen, es versteht nur den datetime Form- Dazu benötigen wir die Funktion to_datetime, um den Objektdatentyp in datetime dtype umzuwandeln.

Die Methode dt.day wird nur den Tag aus dieses Datums extrahieren Die Methode dt.month wird nur den Monat aus dieses Datums extrahieren

[9]:
```
# object spalte zu datetime konvertieren
def change_into_datetime(col):
    train_data[col]= pd.to_datetime(train_data[col])
```

[10]:
```
#train data columns anschauen
train_data.columns
```

[10]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

[11]:
```
for i in ['Date_of_Journey','Dep_Time', 'Arrival_Time']:
    change_into_datetime(i)
```

[12]:
```
# ergebnisse anzeigen
train_data.dtypes
```

[12]:
```
Airline                    object
Date_of_Journey    datetime64[ns]
Source                     object
Destination                object
Route                      object
Dep_Time           datetime64[ns]
```

```
Arrival_Time       datetime64[ns]
Duration                   object
Total_Stops                object
Additional_Info            object
Price                       int64
dtype: object
```

[13]:
```
# tag und monat extrahieren und in zwei spalten (journey_day,journey_mounth)␣
 ↪hinzüfügen
train_data['journey_day']=train_data['Date_of_Journey'].dt.day
train_data['journey_mounth']=train_data['Date_of_Journey'].dt.month
```

[14]:
```
train_data.head()
```

[14]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_mounth
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2022-03-17 22:20:00	2022-03-22 01:10:00	2h 50m	non-stop	No info	3897	24	3
1	Air India	2019-01-05	Kolkata	Banglore	CCU → IXR → BBI → BLR	2022-03-17 05:50:00	2022-03-17 13:15:00	7h 25m	2 stops	No info	7662	5	1
2	Jet Airways	2019-09-06	Delhi	Cochin	DEL → LKO → BOM → COK	2022-03-17 09:25:00	2022-06-10 04:25:00	19h	2 stops	No info	13882	6	9
3	IndiGo	2019-12-05	Kolkata	Banglore	CCU → NAG → BLR	2022-03-17 18:05:00	2022-03-17 23:30:00	5h 25m	1 stop	No info	6218	5	12
4	IndiGo	2019-01-03	Banglore	New Delhi	BLR → NAG → DEL	2022-03-17 16:50:00	2022-03-17 21:35:00	4h 45m	1 stop	No info	13302	3	1

[15]:
```
# data of journey spalte löschen
train_data.drop('Date_of_Journey',axis=1,inplace=True)
```

[16]:
```
train_data.head()
```

[16]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_mounth
0	IndiGo	Banglore	New Delhi	BLR → DEL	2022-03-17 22:20:00	2022-03-22 01:10:00	2h 50m	non-stop	No info	3897	24	3
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2022-03-17 05:50:00	2022-03-17 13:15:00	7h 25m	2 stops	No info	7662	5	1
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2022-03-17 09:25:00	2022-06-10 04:25:00	19h	2 stops	No info	13882	6	9
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	2022-03-17 18:05:00	2022-03-17 23:30:00	5h 25m	1 stop	No info	6218	5	12
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	2022-03-17 16:50:00	2022-03-17 21:35:00	4h 45m	1 stop	No info	13302	3	1

```
             Dep_Time        Arrival_Time Duration Total_Stops  \
0 2022-03-17 22:20:00 2022-03-22 01:10:00   2h 50m    non-stop
1 2022-03-17 05:50:00 2022-03-17 13:15:00   7h 25m     2 stops
2 2022-03-17 09:25:00 2022-06-10 04:25:00      19h     2 stops
3 2022-03-17 18:05:00 2022-03-17 23:30:00   5h 25m      1 stop
4 2022-03-17 16:50:00 2022-03-17 21:35:00   4h 45m      1 stop

  Additional_Info  Price  journey_day  journey_mounth
0         No info   3897           24               3
1         No info   7662            5               1
2         No info  13882            6               9
3         No info   6218            5              12
4         No info  13302            3               1
```

Abflugzeit ist, wenn ein Flugzeug das Gate verlässt. wie bei Date_of_Journey können wir Werte aus Dep_Time extrahieren

[17]:
```
def extract_hour(df,col):
    df[col+"_hour"]=df[col].dt.hour
def extract_min(df,col):
    df[col+"_minute"]=df[col].dt.minute
def drop_column(df,col):
    df.drop(col,axis=1,inplace=True)
```

[18]:
```
#minuten, und stunden aus der spalte 'Dep_time' extrahieren
extract_hour(train_data,'Dep_Time')
extract_min(train_data,'Dep_Time')
# die spalte 'Dep_time' löschen
drop_column(train_data,'Dep_Time')
```

[19]:
```
train_data.head()
```

[19]:
```
       Airline    Source Destination                  Route  \
0       IndiGo  Banglore   New Delhi              BLR → DEL
1    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR
2  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK
3       IndiGo   Kolkata    Banglore        CCU → NAG → BLR
4       IndiGo  Banglore   New Delhi        BLR → NAG → DEL

         Arrival_Time Duration Total_Stops Additional_Info  Price  \
0 2022-03-22 01:10:00   2h 50m    non-stop         No info   3897
1 2022-03-17 13:15:00   7h 25m     2 stops         No info   7662
2 2022-06-10 04:25:00      19h     2 stops         No info  13882
3 2022-03-17 23:30:00   5h 25m      1 stop         No info   6218
4 2022-03-17 21:35:00   4h 45m      1 stop         No info  13302

   journey_day  journey_mounth  Dep_Time_hour  Dep_Time_minute
0           24               3             22               20
```

1	5	1	5	50
2	6	9	9	25
3	5	12	18	5
4	3	1	16	50

Ankunftszeit ist, wenn das Flugzeug am Gate ankommt. wie bei Dep_time können wir Werte aus 'Arrival_time' extrahieren

[20]:
```
# Extracting Hours
extract_hour(train_data,'Arrival_Time')

# Extracting minutes
extract_min(train_data,'Arrival_Time')

# Now we can drop Arrival_Time as it is of no use
drop_column(train_data,'Arrival_Time')
```

[21]:
```
train_data.head()
```

[21]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_mounth	Dep_Time_hour	Dep_Time_minute	Arrival_Time_hour	Arrival_Time_minute
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20	1	10
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	5	50	13	15
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	6	9	9	25	4	25
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	5	12	18	5	23	30
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	3	1	16	50	21	35

0.4 Abgeleitete Merkmale aus Daten extrahieren

[22]:
```
## um später unsere model fütter zu können
## ist notwendig paar spalten in deselbe format umzuwandeln ---> später stunden␣
 ↪und minuten extrahieren
duration = list(train_data['Duration'])
```

[23]:
```
x='2h 50m'
len(x.split(' '))
```

[23]: 2

Die spalte 'Duration' vorbereiten ——> 1) alle spalte element in desselbe form umwandeln Z.b '2h 0m' ——> 2) stunde und minute extrahieren Z.B '2h 0m' => 2 - 0

[24]:
```
# 1/
for i in range(len(duration)):
    if len(duration[i].split(' '))==2:
        pass
    else:
        if 'h' in duration[i]:
            duration[i]=duration[i]+' 0m'
        else:
            duration[i]='0h '+duration[i]
```

[25]:
```
# die spalte Duration umwandeln
train_data['Duration']=duration
```

[26]:
```
train_data.tail()
```

[26]:

	Airline	Source	Destination	Route	Duration	\
10678	Air Asia	Kolkata	Banglore	CCU → BLR	2h 30m	
10679	Air India	Kolkata	Banglore	CCU → BLR	2h 35m	
10680	Jet Airways	Banglore	Delhi	BLR → DEL	3h 0m	
10681	Vistara	Banglore	New Delhi	BLR → DEL	2h 40m	
10682	Air India	Delhi	Cochin	DEL → GOI → BOM → COK	8h 20m	

	Total_Stops	Additional_Info	Price	journey_day	journey_mounth	\
10678	non-stop	No info	4107	4	9	
10679	non-stop	No info	4145	27	4	
10680	non-stop	No info	7229	27	4	
10681	non-stop	No info	12648	3	1	
10682	2 stops	No info	11753	5	9	

	Dep_Time_hour	Dep_Time_minute	Arrival_Time_hour	Arrival_Time_minute
10678	19	55	22	25
10679	20	45	23	20
10680	8	20	11	20
10681	11	30	14	10
10682	10	55	19	15

[27]:
```
from pytimeparse.timeparse import timeparse # um die zahl in sekunde zu rechnen!
```

[28]:
```
'2h 35m'.split(' ')[1][0:-1]
```

[28]:
```
'35'
```

[29]:
```
# 2/
def hour(x):
    return x.split(' ')[0][0:-1]

def minute(x):
    return x.split(' ')[1][0:-1]
```

[30]:
```
# die stunde und minute trennen
train_data['Duration_hours']=train_data['Duration'].apply(hour)
train_data['Duration_mins']=train_data['Duration'].apply(minute)
```

[31]:
```
train_data.head()
```

[31]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_mounth	Dep_Time_hour	Dep_Time_minute	Arrival_Time_hour	Arrival_Time_minute	Duration_hours	Duration_mins
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20	1	10	2	50
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	1	5	50	13	15	7	25
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	6	9	9	25	4	25	19	0
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	5	12	18	5	23	30	5	25
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	3	1	16	50	21	35	4	45

[32]:
```
# die spalte Duration löschen , nicht mehr gebrauchbar ----
drop_column(train_data,'Duration')
```

[33]:
```
train_data.head()
```

[33]:
```
       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  journey_day  journey_mounth  Dep_Time_hour  \
0         No info   3897           24               3             22
1         No info   7662            5               1              5
2         No info  13882            6               9              9
3         No info   6218            5              12             18
4         No info  13302            3               1             16

   Dep_Time_minute  Arrival_Time_hour  Arrival_Time_minute Duration_hours  \
0               20                  1                   10              2
1               50                 13                   15              7
2               25                  4                   25             19
3                5                 23                   30              5
4               50                 21                   35              4

  Duration_mins
0            50
1            25
2             0
3            25
4            45
```

[34]:
```
# Daten typ unsere DataFrame anzeigen
train_data.dtypes
```

[34]:
```
Airline               object
Source                object
Destination           object
Route                 object
Total_Stops           object
Additional_Info       object
Price                  int64
journey_day            int64
journey_mounth         int64
Dep_Time_hour          int64
Dep_Time_minute        int64
Arrival_Time_hour      int64
Arrival_Time_minute    int64
Duration_hours        object
Duration_mins         object
dtype: object
```

[35]:
```
# die 'Duration_hours' und 'Duration_mins ' zu int umwandeln
train_data['Duration_hours']=train_data['Duration_hours'].astype(int)
train_data['Duration_mins']=train_data['Duration_mins'].astype(int)
```

———Umgang mit kategorialen Daten——— ich verwende 2 Codierungs Techniken , um kategoriale Daten in ein numerisches Format umzuwandeln: ***Nominale Daten –> Daten sind nicht in beliebiger Reihenfolge –> OneHotEncoder wird in diesem Fall verwendet*** Ordinale Daten –> Daten sind in Ordnung –> LabelEncoder wird in diesem Fall verwendet

[36]:
```
#Daten type der DataFrame anzeigen
train_data.dtypes
```

[36]:
```
Airline                object
Source                 object
Destination            object
Route                  object
Total_Stops            object
Additional_Info        object
Price                   int64
journey_day             int64
journey_mounth          int64
Dep_Time_hour           int64
Dep_Time_minute         int64
Arrival_Time_hour       int64
Arrival_Time_minute     int64
Duration_hours          int32
Duration_mins           int32
dtype: object
```

[37]:
```
for i in train_data.dtypes:
    print(i=='O')
```

```
True
True
True
True
True
True
False
False
False
False
False
False
False
False
False
```

[38]:
```
# die spalten in zwei daten typ zerlegen :
cat_col = [col for col in train_data.columns if train_data[col].dtypes=='O' ] #␣
 ↪katorigale Merkmale
cat_col
```

[38]:
```
['Airline', 'Source', 'Destination', 'Route', 'Total_Stops', 'Additional_Info']
```

[39]:
```
cont_col = [col for col in train_data.columns if train_data[col].dtypes!='O'] #␣
 ↪kontuierlische Merkmale
cont_col
```

[39]:
```
['Price',
 'journey_day',
 'journey_mounth',
 'Dep_Time_hour',
 'Dep_Time_minute',
 'Arrival_Time_hour',
 'Arrival_Time_minute',
 'Duration_hours',
 'Duration_mins']
```

[40]:
```
train_data.head()
# norminal Data -- Onehot
####For categorical variables where no ordinal relationship exists,
####the integer encoding may not be enough, at best, or misleading to the model␣
 ↪at worst.
# ordinal data label encoder
#####In ordinal encoding, each unique category value is assigned an integer␣
 ↪value.
#####For example, "red" is 1, "green" is 2, and "blue" is 3.
```

[40]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	journey_day	journey_mounth	Dep_Time_hour	Dep_Time_minute	Arrival_Time_hour	Arrival_Time_minute	Duration_hours	Duration_mins
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	3	22	20	1	10	2	50
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	5	1	5	50	13	15	7	25
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	6	9	9	25	4	25	19	0
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	6218	5	12	18	5	23	30	5	25
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	13302	3	1	16	50	21	35	4	45

```
1               50                13                15                 7
2               25                 4                25                19
3                5                23                30                 5
4               50                21                35                 4

   Duration_mins
0             50
1             25
2              0
3             25
4             45
```

[41]:
```
categorical=train_data[cat_col]
categorical.head()
```

[41]:
```
       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
0         No info
1         No info
2         No info
3         No info
4         No info
```

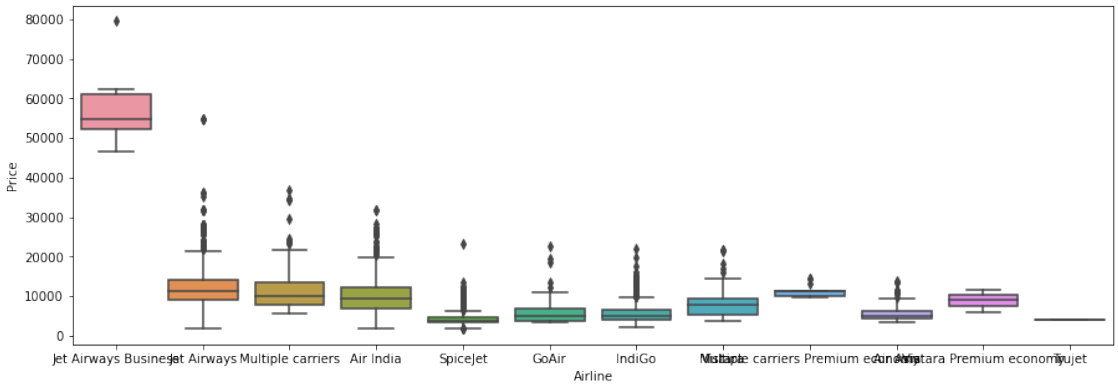
[42]:
```
categorical['Airline'].value_counts()
```

[42]:
```
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
```

[43]:
```
# boxplot von (Airline und Price ) dastellen
plt.figure(figsize=(15,5))
```

```
sns.boxplot(x='Airline',y='Price',data=train_data.
 ↪sort_values('Price',ascending=False))
```

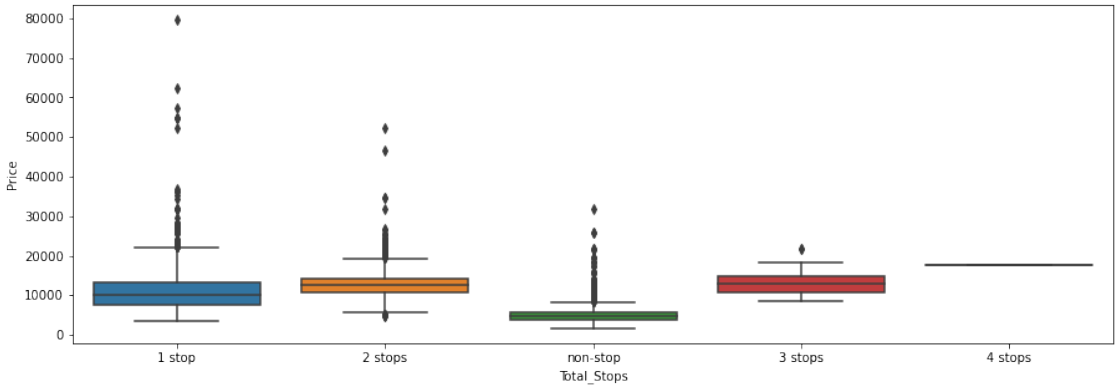
[43]: <AxesSubplot:xlabel='Airline', ylabel='Price'>



0.4.1 Fazit –> Aus dem Diagramm können wir sehen, dass Jet Airways Business den höchsten Preis hat. Abgesehen von der ersten Fluggesellschaft haben fast alle einen ähnlichen Median

[44]:
```
#Boxplot von Total_Stops und Price Darstellen
plt.figure(figsize=(15,5))
sns.boxplot(x='Total_Stops',y='Price',data=train_data.
 ↪sort_values('Price',ascending=False))
```

[44]: <AxesSubplot:xlabel='Total_Stops', ylabel='Price'>



[45]:
```
# Da es sich bei der Fluggesellschaft um nominale kategoriale Daten handelt,
#führen wir One Hot Encoding durch
Airline = pd.get_dummies(categorical['Airline'],drop_first=True)
```

[46]:
```
Airline.head()
```

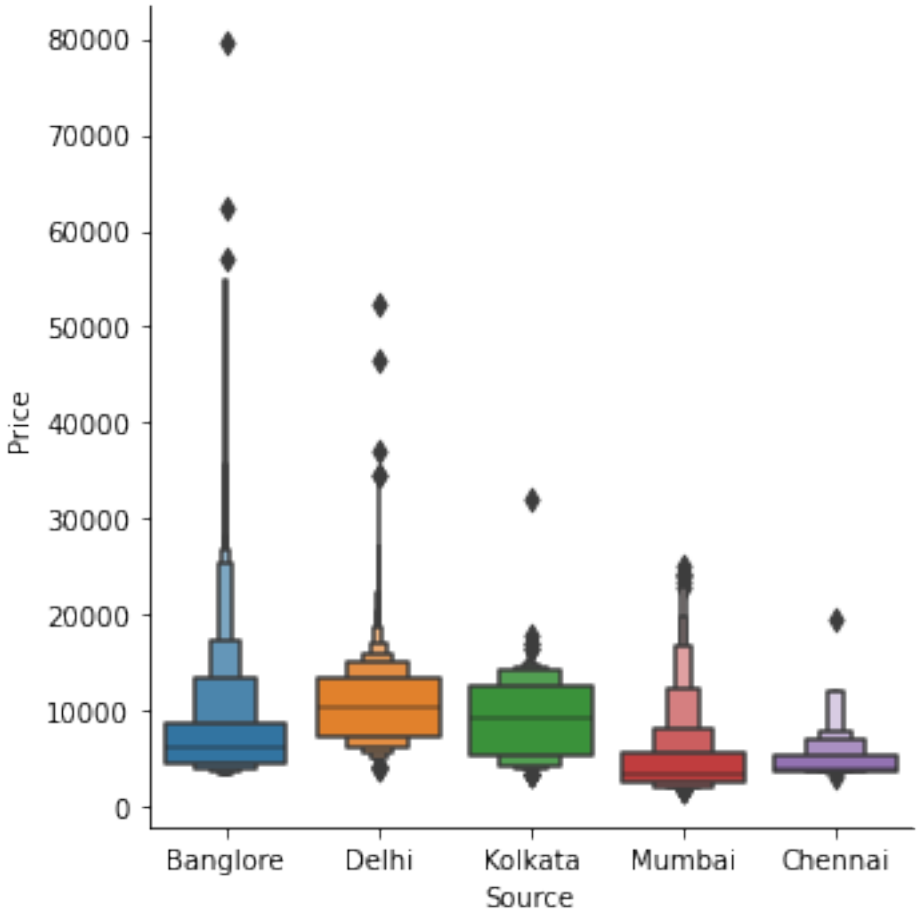
[46]:
```
   Air India  GoAir  IndiGo  Jet Airways  Jet Airways Business  \
0          0      0       1            0                     0
1          1      0       0            0                     0
2          0      0       0            1                     0
3          0      0       1            0                     0
4          0      0       1            0                     0

   Multiple carriers  Multiple carriers Premium economy  SpiceJet  Trujet  \
0                  0                                  0         0       0
1                  0                                  0         0       0
2                  0                                  0         0       0
3                  0                                  0         0       0
4                  0                                  0         0       0

   Vistara  Vistara Premium economy
0        0                        0
1        0                        0
2        0                        0
3        0                        0
4        0                        0
```

[47]:
```
categorical['Source'].value_counts()
```

[47]:
```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

[120]:
```
# Source vs Price

plt.figure(figsize=(15,5))
sns.catplot(x='Source',y='Price',data=train_data.
 ↪sort_values('Price',ascending=False),kind='boxen')
```

[120]:
```
<seaborn.axisgrid.FacetGrid at 0x1c7c2de1e50>

<Figure size 1080x360 with 0 Axes>
```



[49]:
```
# Da es sich bei der Quelle um nominale kategoriale Daten handelt,
# führen wir One Hot Encoding durch
Source = pd.get_dummies(categorical['Source'],drop_first=True)
Source.head()
```

[49]:

	Chennai	Delhi	Kolkata	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

[50]:
```
Source.shape
```

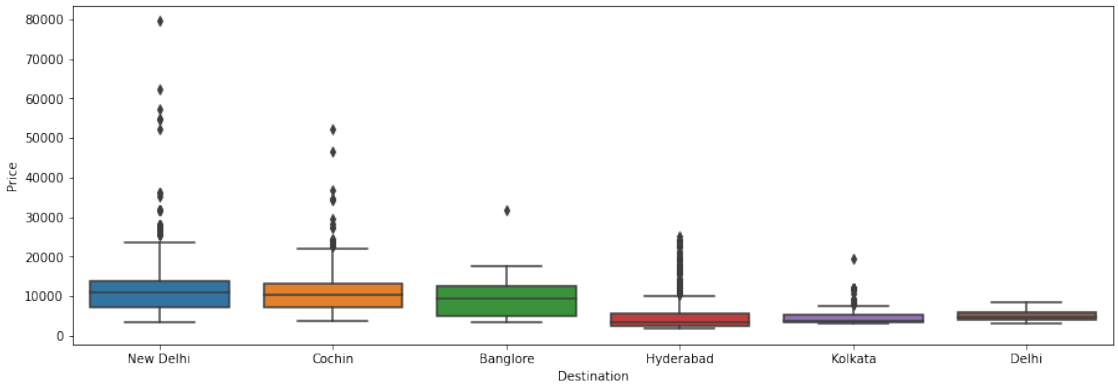
[50]:
```
(10682, 4)
```

[51]:
```
categorical['Destination'].value_counts()
```

[51]:
```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

[52]:
```
plt.figure(figsize=(15,5))
sns.boxplot(x='Destination',y='Price',data=train_data.
 ↪sort_values('Price',ascending=False))
```

[52]:
```
<AxesSubplot:xlabel='Destination', ylabel='Price'>
```



[53]:
```
# norminal Data -- Onehot
##
# Da das 'Destination' nominale kategoriale Daten sind,
# führen wir One Hot Encoding durch
## ['Airline', 'Source', 'Destination', 'Route', 'Total_Stops',␣
 ↪'Additional_Info']
Destination = pd.get_dummies(categorical['Destination'],drop_first=True)
Destination.head()
```

[53]:

	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
0	0	0	0	0	1
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1

[54]:
```
cat = categorical.copy()
cat
```

[54]:
```
            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
…              …       …          …                     …          …
10678     Air Asia   Kolkata    Banglore              CCU → BLR    non-stop
10679    Air India   Kolkata    Banglore              CCU → BLR    non-stop
10680  Jet Airways  Banglore       Delhi              BLR → DEL    non-stop
10681      Vistara  Banglore   New Delhi              BLR → DEL    non-stop
10682    Air India     Delhi      Cochin  DEL → GOI → BOM → COK     2 stops

      Additional_Info
0             No info
1             No info
2             No info
3             No info
4             No info
…               …
10678         No info
10679         No info
10680         No info
10681         No info
10682         No info

[10682 rows x 6 columns]
```

[55]:
```
categorical['Route']
```

[55]:
```
0                   BLR → DEL
1        CCU → IXR → BBI → BLR
2        DEL → LKO → BOM → COK
3              CCU → NAG → BLR
4              BLR → NAG → DEL
                 …
10678               CCU → BLR
10679               CCU → BLR
10680               BLR → DEL
10681               BLR → DEL
10682    DEL → GOI → BOM → COK
Name: Route, Length: 10682, dtype: object
```

[56]:
```
categorical['Route_1']= categorical['Route'].str.split('→').str[0]
categorical['Route_2']= categorical['Route'].str.split('→').str[1]
categorical['Route_3']= categorical['Route'].str.split('→').str[2]
categorical['Route_4']= categorical['Route'].str.split('→').str[4]
```

```
categorical['Route_5']= categorical['Route'].str.split('→').str[5]
```

[57]:
```
 categorical['Route_5']
```

[57]:
```
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
         …
10678     NaN
10679     NaN
10680     NaN
10681     NaN
10682     NaN
Name: Route_5, Length: 10682, dtype: object
```

[58]:
```
# route Spalte löschen
drop_column(categorical,'Route')
```

[59]:
```
categorical.isnull().sum()
```

[59]:
```
Airline               0
Source                0
Destination           0
Total_Stops           0
Additional_Info       0
Route_1               0
Route_2               0
Route_3            3491
Route_4           10636
Route_5           10681
dtype: int64
```

[60]:
```
categorical.columns
```

[60]:
```
Index(['Airline', 'Source', 'Destination', 'Total_Stops', 'Additional_Info',
       'Route_1', 'Route_2', 'Route_3', 'Route_4', 'Route_5'],
      dtype='object')
```

[61]:
```
# die leer werte stellen mit none erzetsen
for i in ['Route_3','Route_4','Route_5']:
    categorical[i].fillna('None',inplace=True)
```

[62]:
```
categorical.isnull().sum()
```

[62]:
```
Airline            0
Source             0
Destination        0
Total_Stops        0
Additional_Info    0
Route_1            0
Route_2            0
Route_3            0
Route_4            0
Route_5            0
dtype: int64
```

[63]:
```
#extrahieren wie viele Kategorien in jedem spalte
for i in categorical.columns:
    print(f'{i} has total {len(categorical[i].value_counts())} categories')
```

```
Airline has total 12 categories
Source has total 5 categories
Destination has total 6 categories
Total_Stops has total 5 categories
Additional_Info has total 10 categories
Route_1 has total 5 categories
Route_2 has total 45 categories
Route_3 has total 30 categories
Route_4 has total 6 categories
Route_5 has total 2 categories
```

[ ]:
```
###  wir haben viele Funktionen in Route ,
### eine Hot-Encoding wird keine bessere Option sein,
### ==>  Label Encoding anwenden
```

[64]:
```
from sklearn.preprocessing import LabelEncoder
```

[65]:
```
encoder=LabelEncoder()
```

[66]:
```
categorical.columns
```

[66]:
```
Index(['Airline', 'Source', 'Destination', 'Total_Stops', 'Additional_Info',
       'Route_1', 'Route_2', 'Route_3', 'Route_4', 'Route_5'],
      dtype='object')
```

[67]:
```
for i in ['Route_1','Route_2','Route_3','Route_4','Route_5']:
    categorical[i]=encoder.fit_transform(categorical[i])
```

[68]:
```
categorical.head()
```

[68]:
```
        Airline    Source Destination Total_Stops Additional_Info  Route_1  \
0        IndiGo  Banglore   New Delhi    non-stop         No info        0
```

```
1    Air India   Kolkata    Banglore     2 stops         No info        2
2  Jet Airways     Delhi      Cochin     2 stops         No info        3
3       IndiGo   Kolkata    Banglore      1 stop         No info        2
4       IndiGo  Banglore   New Delhi      1 stop         No info        0

   Route_2  Route_3  Route_4  Route_5
0       13       29        5        1
1       25        1        5        1
2       32        4        5        1
3       34        3        5        1
4       34        8        5        1
```

[69]:
```
categorical['Additional_Info'].value_counts()
```

[69]:
```
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
Red-eye flight                     1
2 Long layover                     1
Name: Additional_Info, dtype: int64
```

[70]:
```
## Additional_Info enthält fast 80 % no_info,
## also können wir diese Spalte weglassen
drop_column(categorical,'Additional_Info')
```

[71]:
```
categorical['Total_Stops'].unique()
```

[71]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

[72]:
```
##  es geht um  ordinalen kategorialen Typs ist,
## => LabelEncoder führen
dict={'non-stop':0, '2 stops':1, '1 stop':2, '3 stops':3, '4 stops':4}
```

[73]:
```
# die spalte total_stops mit zahlen ersetzen
categorical['Total_Stops']=categorical['Total_Stops'].map(dict)
```

[74]:
```
categorical.head()
```

[74]:
```
       Airline    Source Destination  Total_Stops  Route_1  Route_2  Route_3  \
0       IndiGo  Banglore   New Delhi            0        0       13       29
1    Air India   Kolkata    Banglore            1        2       25        1
2  Jet Airways     Delhi      Cochin            1        3       32        4
```

```
3       IndiGo   Kolkata    Banglore            2        2       34        3
4       IndiGo  Banglore   New Delhi            2        0       34        8

   Route_4  Route_5
0        5        1
1        5        1
2        5        1
3        5        1
4        5        1
```

[121]:
```
# Concatenate dataframe --> categorical + Airline + Source + Destination
data_train= pd.concat([categorical,Airline,Source,Destination,\
                       train_data[cont_col]],axis=1)
```

[122]:
```
data_train
```

[122]:
```
           Airline    Source Destination  Total_Stops  Route_1  Route_2  \
0           IndiGo  Banglore   New Delhi            0        0       13
1        Air India   Kolkata    Banglore            1        2       25
2      Jet Airways     Delhi      Cochin            1        3       32
3           IndiGo   Kolkata    Banglore            2        2       34
4           IndiGo  Banglore   New Delhi            2        0       34
…              …       …          …          …      …      …
10678     Air Asia   Kolkata    Banglore            0        2        5
10679    Air India   Kolkata    Banglore            0        2        5
10680  Jet Airways  Banglore       Delhi            0        0       13
10681      Vistara  Banglore   New Delhi            0        0       13
10682    Air India     Delhi      Cochin            1        3       16

       Route_3  Route_4  Route_5  Air India  GoAir  IndiGo  Jet Airways  \
0           29        5        1          0      0       1            0
1            1        5        1          1      0       0            0
2            4        5        1          0      0       0            1
3            3        5        1          0      0       1            0
4            8        5        1          0      0       1            0
…          …      …        …          …    …      …          …
10678       29        5        1          0      0       0            0
10679       29        5        1          1      0       0            0
10680       29        5        1          0      0       0            1
10681       29        5        1          0      0       0            0
10682        4        5        1          1      0       0            0

       Jet Airways Business  Multiple carriers  \
0                         0                  0
1                         0                  0
2                         0                  0
3                         0                  0
```

```
4                          0                  0
…                        …                  …
10678                      0                  0
10679                      0                  0
10680                      0                  0
10681                      0                  0
10682                      0                  0

       Multiple carriers Premium economy  SpiceJet  …  Delhi  Kolkata  \
0                                      0         0  …      0        0
1                                      0         0  …      0        1
2                                      0         0  …      1        0
3                                      0         0  …      0        1
4                                      0         0  …      0        0
…                                    …       …  …    …      …
10678                                  0         0  …      0        1
10679                                  0         0  …      0        1
10680                                  0         0  …      0        0
10681                                  0         0  …      0        0
10682                                  0         0  …      1        0

       Mumbai  Cochin  Delhi  Hyderabad  Kolkata  New Delhi  Price  \
0           0       0      0          0        0          1   3897
1           0       0      0          0        0          0   7662
2           0       1      0          0        0          0  13882
3           0       0      0          0        0          0   6218
4           0       0      0          0        0          1  13302
…       …      …    …         …       …         …    …
10678       0       0      0          0        0          0   4107
10679       0       0      0          0        0          0   4145
10680       0       0      1          0        0          0   7229
10681       0       0      0          0        0          1  12648
10682       0       1      0          0        0          0  11753

       journey_day  journey_mounth  Dep_Time_hour  Dep_Time_minute  \
0               24               3             22               20
1                5               1              5               50
2                6               9              9               25
3                5              12             18                5
4                3               1             16               50
…              …               …            …               …
10678            4               9             19               55
10679           27               4             20               45
10680           27               4              8               20
10681            3               1             11               30
10682            5               9             10               55
```

```
       Arrival_Time_hour  Arrival_Time_minute  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
…                    …                     …                …              …
10678                 22                   25               2             30
10679                 23                   20               2             35
10680                 11                   20               3              0
10681                 14                   10               2             40
10682                 19                   15               8             20

[10682 rows x 38 columns]
```

[77]:
```
# die spalten Airlaine Source und Destination löschen
drop_column(data_train,'Airline')
drop_column(data_train,'Source')
drop_column(data_train,'Destination')
```

[78]:
```
data_train.head(2)
```

[78]:
```
   Total_Stops  Route_1  Route_2  Route_3  Route_4  Route_5  Air India  GoAir  \
0            0        0       13       29        5        1          0      0
1            1        2       25        1        5        1          1      0

   IndiGo  Jet Airways  …  New Delhi  Price  journey_day  journey_mounth  \
0       1            0  …          1   3897           24               3
1       0            0  …          0   7662            5               1

   Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  Arrival_Time_minute  \
0             22               20                  1                   10
1              5               50                 13                   15

   Duration_hours  Duration_mins
0               2             50
1               7             25

[2 rows x 35 columns]
```

[79]:
```
pd.set_option('display.max_columns',35)
data_train.head(3)
```

[79]:
```
   Total_Stops  Route_1  Route_2  Route_3  Route_4  Route_5  Air India  GoAir  \
0            0        0       13       29        5        1          0      0
1            1        2       25        1        5        1          1      0
2            1        3       32        4        5        1          0      0
```

```
   IndiGo  Jet Airways  Jet Airways Business  Multiple carriers  \
0       1            0                     0                  0
1       0            0                     0                  0
2       0            1                     0                  0

   Multiple carriers Premium economy  SpiceJet  Trujet  Vistara  \
0                                  0         0       0        0
1                                  0         0       0        0
2                                  0         0       0        0

   Vistara Premium economy  Chennai  Delhi  Kolkata  Mumbai  Cochin  Delhi  \
0                        0        0      0        0       0       0      0
1                        0        0      0        1       0       0      0
2                        0        0      1        0       0       1      0

   Hyderabad  Kolkata  New Delhi  Price  journey_day  journey_mounth  \
0          0        0          1   3897           24               3
1          0        0          0   7662            5               1
2          0        0          0  13882            6               9

   Dep_Time_hour  Dep_Time_minute  Arrival_Time_hour  Arrival_Time_minute  \
0             22               20                  1                   10
1              5               50                 13                   15
2              9               25                  4                   25

   Duration_hours  Duration_mins
0               2             50
1               7             25
2              19              0
```

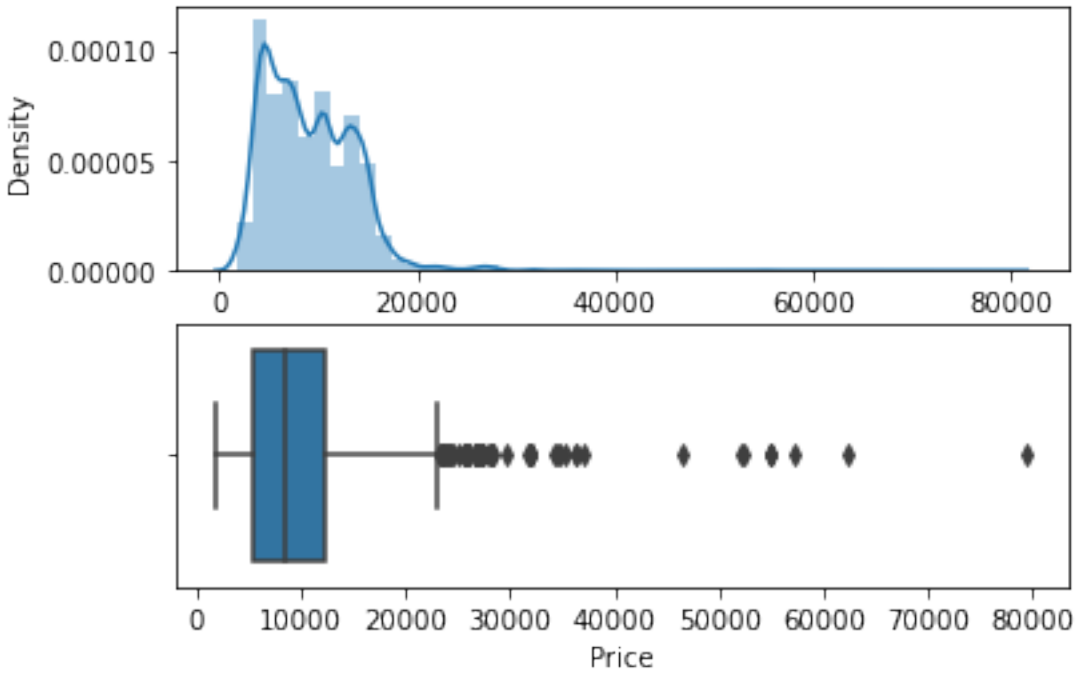
1 outlier detektion

[80]:
```
def plot(df,col):
    fig,(ax1,ax2)=plt.subplots(2,1)
    sns.distplot(df[col],ax=ax1)
    sns.boxplot(df[col],ax=ax2)
```

[81]:
```
plot(data_train,'Price')
```

```
C:\Users\khale\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\khale\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
```

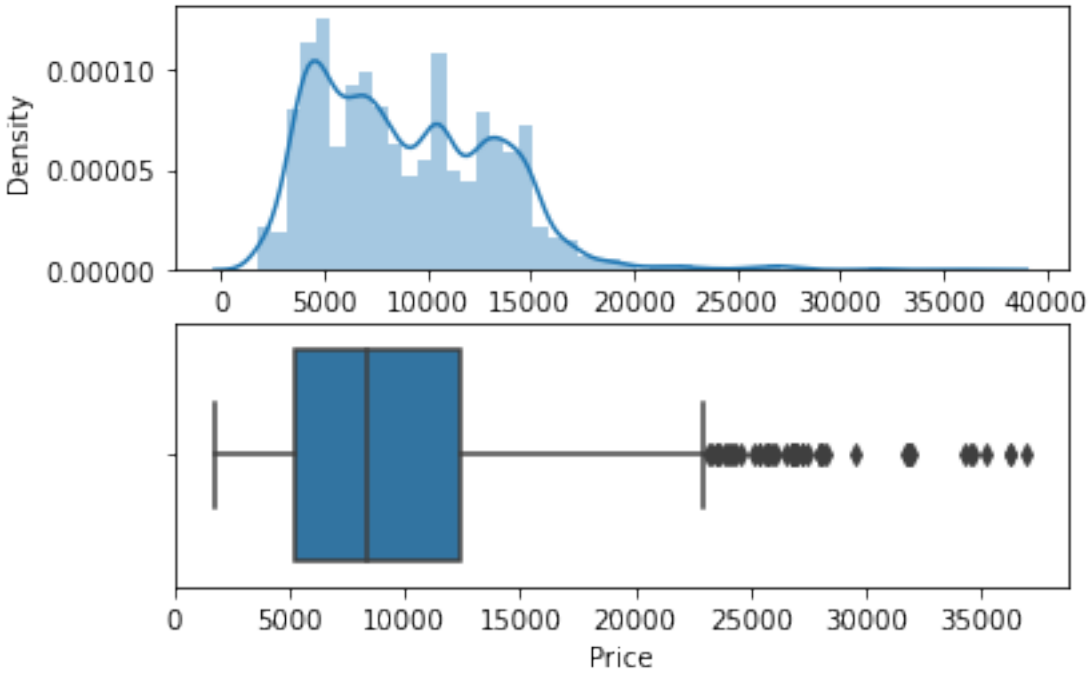
```
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[82]:
```
# outlier löschen
data_train['Price']= np.where(data_train['Price']>=40000,data_train['Price'].
 ↪median(),data_train['Price'])
```

[83]:
```
plot(data_train,'Price')
```

```
C:\Users\khale\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\khale\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[84]:
```
### Trennen Sie Ihre unabhängigen und abhängigen Daten

X = data_train.drop('Price',axis=1)
X.head()
```

[84]:

	Total_Stops	Route_1	Route_2	Route_3	Route_4	Route_5	Air India	GoAir
0	0	0	13	29	5	1	0	0
1	1	2	25	1	5	1	1	0
2	1	3	32	4	5	1	0	0
3	2	2	34	3	5	1	0	0
4	2	0	34	8	5	1	0	0

	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers
0	1	0	0	0
1	0	0	0	0
2	0	1	0	0
3	1	0	0	0
4	1	0	0	0

	Multiple carriers Premium economy	SpiceJet	Trujet	Vistara
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

```
4                                    0         0       0         0

   Vistara Premium economy  Chennai  Delhi  Kolkata  Mumbai  Cochin  Delhi  \
0                        0        0      0        0       0       0      0
1                        0        0      0        1       0       0      0
2                        0        0      1        0       0       1      0
3                        0        0      0        1       0       0      0
4                        0        0      0        0       0       0      0

   Hyderabad  Kolkata  New Delhi  journey_day  journey_mounth  Dep_Time_hour  \
0          0        0          1           24               3             22
1          0        0          0            5               1              5
2          0        0          0            6               9              9
3          0        0          0            5              12             18
4          0        0          1            3               1             16

   Dep_Time_minute  Arrival_Time_hour  Arrival_Time_minute  Duration_hours  \
0               20                  1                   10               2
1               50                 13                   15               7
2               25                  4                   25              19
3                5                 23                   30               5
4               50                 21                   35               4

   Duration_mins
0             50
1             25
2              0
3             25
4             45
```

[85]:
```
X.shape
```

[85]:
```
(10682, 34)
```

[86]:
```
y = data_train['Price']
y
```

[86]:
```
0         3897.0
1         7662.0
2        13882.0
3         6218.0
4        13302.0
          ...
10678     4107.0
10679     4145.0
10680     7229.0
10681    12648.0
```

```
10682    11753.0
Name: Price, Length: 10682, dtype: float64
```

[126]:
```
print(type(X),type(y))
```

```
<class 'pandas.core.frame.DataFrame'> <class 'pandas.core.series.Series'>
```

[128]:
```
X.isnull().sum()
```

[128]:
```
<bound method NDFrame.head of Total_Stops                         0
Route_1                               0
Route_2                               0
Route_3                               0
Route_4                               0
Route_5                               0
Air India                             0
GoAir                                 0
IndiGo                                0
Jet Airways                           0
Jet Airways Business                  0
Multiple carriers                     0
Multiple carriers Premium economy     0
SpiceJet                              0
Trujet                                0
Vistara                               0
Vistara Premium economy               0
Chennai                               0
Delhi                                 0
Kolkata                               0
Mumbai                                0
Cochin                                0
Delhi                                 0
Hyderabad                             0
Kolkata                               0
New Delhi                             0
journey_day                           0
journey_mounth                        0
Dep_Time_hour                         0
Dep_Time_minute                       0
Arrival_Time_hour                     0
Arrival_Time_minute                   0
Duration_hours                        0
Duration_mins                         0
dtype: int64>
```

[129]:
```
y.isnull().sum()
```

[129]:
```
0
```

[ ]:
```
# ==> es gibt kein fehlende Wert,!
```

[87]:
```
#  Informationswerte oder eine Matrix finden,
# um die Beziehung zwischen allen Merkmalen zu finden.
from sklearn.feature_selection import mutual_info_classif
```

[88]:
```
mutual_info_classif(X,y)
```

[88]:
```
array([2.14445435, 2.04317822, 2.75736685, 2.31766745, 0.7376523 ,
       1.79386812, 0.76595518, 0.08899554, 0.6781991 , 0.94479075,
       0.01762642, 0.55555379, 0.02410563, 0.32835272, 0.00975017,
       0.23862649, 0.        , 0.18115241, 1.55115335, 0.86935597,
       0.29076484, 1.53857209, 0.40495301, 0.28738162, 0.17534013,
       0.36709468, 1.11547901, 0.84732007, 1.4537781 , 1.21848699,
       1.8508155 , 1.53057194, 1.7914488 , 1.05969602])
```

[89]:
```
imp = pd.DataFrame(mutual_info_classif(X,y),index=X.columns)
imp
```

[89]:

	0
Total_Stops	2.120111
Route_1	2.047357
Route_2	2.787160
Route_3	2.296782
Route_4	0.721625
Route_5	1.772757
Air India	0.764172
GoAir	0.111924
IndiGo	0.684993
Jet Airways	0.915635
Jet Airways Business	0.000000
Multiple carriers	0.549740
Multiple carriers Premium economy	0.022855
SpiceJet	0.312998
Trujet	0.004961
Vistara	0.222607
Vistara Premium economy	0.000378
Chennai	0.183276
Delhi	1.529027
Kolkata	0.886552
Mumbai	0.288735
Cochin	1.557391
Delhi	0.420911
Hyderabad	0.283136
Kolkata	0.160090
New Delhi	0.368831
journey_day	1.077733

```
journey_mounth                    0.799598
Dep_Time_hour                     1.430939
Dep_Time_minute                   1.216594
Arrival_Time_hour                 1.847963
Arrival_Time_minute               1.541278
Duration_hours                    1.770355
Duration_mins                     1.067507
```

[90]:
```
imp.columns=['importance']
imp.sort_values(by='importance',ascending=False)
```

[90]:

	importance
Route_2	2.787160
Route_3	2.296782
Total_Stops	2.120111
Route_1	2.047357
Arrival_Time_hour	1.847963
Route_5	1.772757
Duration_hours	1.770355
Cochin	1.557391
Arrival_Time_minute	1.541278
Delhi	1.529027
Dep_Time_hour	1.430939
Dep_Time_minute	1.216594
journey_day	1.077733
Duration_mins	1.067507
Jet Airways	0.915635
Kolkata	0.886552
journey_mounth	0.799598
Air India	0.764172
Route_4	0.721625
IndiGo	0.684993
Multiple carriers	0.549740
Delhi	0.420911
New Delhi	0.368831
SpiceJet	0.312998
Mumbai	0.288735
Hyderabad	0.283136
Vistara	0.222607
Chennai	0.183276
Kolkata	0.160090
GoAir	0.111924
Multiple carriers Premium economy	0.022855
Trujet	0.004961
Vistara Premium economy	0.000378
Jet Airways Business	0.000000

[91]:
```
from sklearn.model_selection import train_test_split
```

[92]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

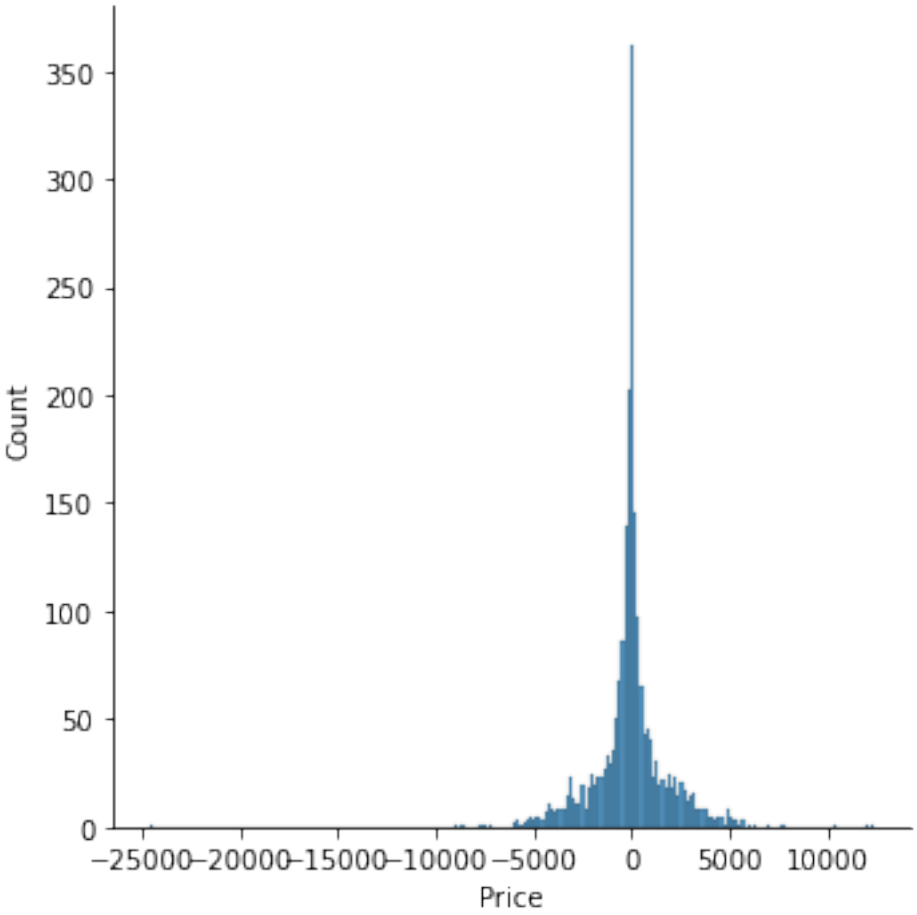
[93]:
```
from sklearn import metrics
```

[94]:
```
def predict(ml_model):
    model = ml_model.fit(X_train,y_train)
    print(f'Training score: {model.score(X_train,y_train)}')
    y_prediction=model.predict(X_test)
    print(f'predictions are:\n {y_prediction}')
    print('\n')

    r2_score=metrics.r2_score(y_test,y_prediction)
    print(f'r2 score is:{r2_score}')

    print('MAE: ', metrics.mean_absolute_error(y_test,y_prediction))
    print('MSE: ', metrics.mean_squared_error(y_test,y_prediction))
    print('RMSE: ',np.sqrt(metrics.mean_squared_error(y_test,y_prediction)))

    sns.displot(y_test-y_prediction)
```

[95]:
```
from sklearn.ensemble import RandomForestRegressor
```

[96]:
```
predict(RandomForestRegressor())
```

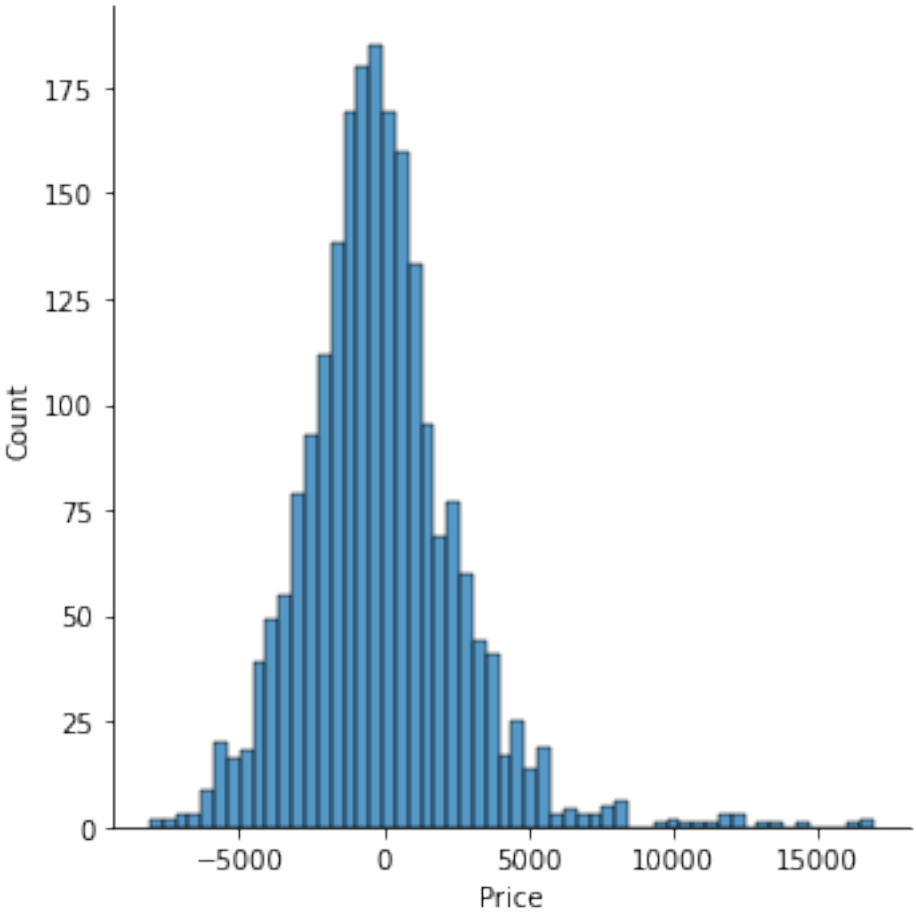
```
Training score: 0.9555534919222949
predictions are:
 [12695.505      13578.95533333  8712.05       …  3603.276
  8571.18        4672.2       ]


r2 score is:0.8052560554813359
MAE:  1138.4285246726574
MSE:  3531818.26973455
RMSE:  1879.3132441757946
```



[97]:
```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
```

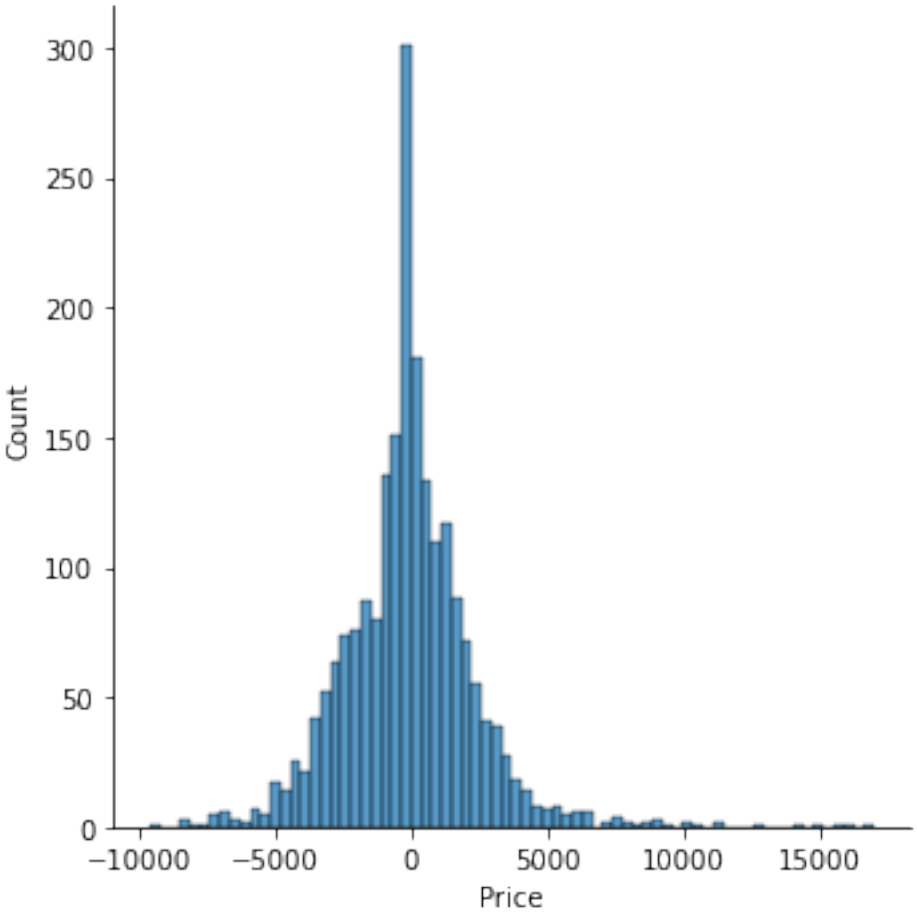
[98]:
```
predict(LinearRegression())
```

```
Training score: 0.5918372122618918
predictions are:
 [12145.63501248 10698.08389202  8412.03510388 …  3638.06651841
  9289.15146591  5889.8409409 ]


r2 score is:0.5994398674288917
MAE:  1970.2455792597102
MSE:  7264439.455812447
RMSE:  2695.262409453381
```



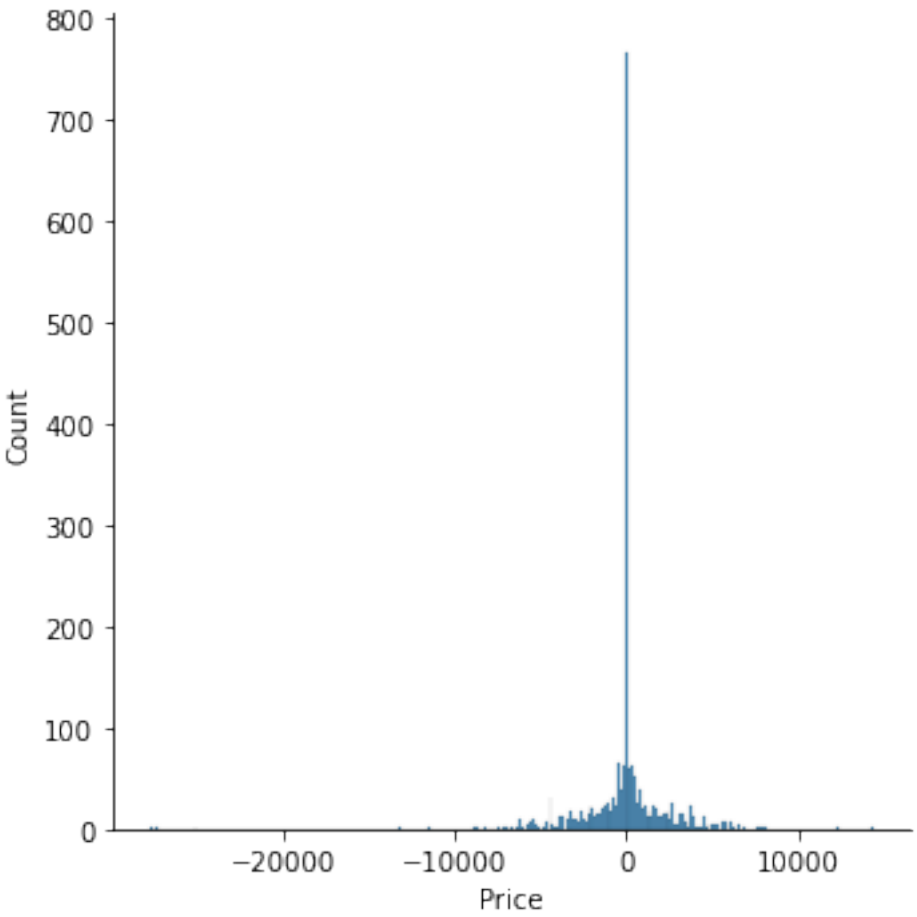
[99]:
```
predict(KNeighborsRegressor())
```

```
Training score: 0.7713174476597382
predictions are:
 [12213.2 10839.2  8757.2 …  6951.   8799.   4928.2]


r2 score is:0.6707803148662894
MAE:  1687.2852597098738
MSE:  5970630.314515676
RMSE:  2443.4873264487533
```



[100]:
```
predict(DecisionTreeRegressor())
```

```
Training score: 0.9686328386654128
predictions are:
 [14151. 13832.  8534. …  3597.  8610.  4668.]


r2 score is:0.6656711918706972
MAE:  1308.2096084854156
MSE:  6063287.849941246
RMSE:  2462.374433334875
```



[101]:
```
from sklearn.ensemble import RandomForestRegressor
```

[102]:
```
reg_rf = RandomForestRegressor()
```

[103]:
```
from sklearn.model_selection import RandomizedSearchCV
```

[ ]:
```

```

[104]:
```
n_estimators=[int(x) for x in np.linspace(start=100,stop=1200,num=6)]
max_depth=[int(x) for x in np.linspace(start=5,stop=30,num=4)]
```

[105]:
```
RandomForestRegressor()
```

[105]:
```
RandomForestRegressor()
```

[ ]:
```

```

[106]:
```
random_grid={
    'n_estimators': n_estimators,
    'max_features':['auto','sqrt'],
    'max_depth':max_depth,
    'min_samples_split':[5,10,14,100]
}
```

[107]:
```
rf_random =␣
↪RandomizedSearchCV(estimator=reg_rf,param_distributions=random_grid,cv=3,verbose=2,n_jobs=-
```

[108]:
```
rf_random.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

[108]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [5, 13, 21, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_split': [5, 10, 14, 100],
                                        'n_estimators': [100, 320, 540, 760,
                                                         980, 1200]},
                   verbose=2)
```

[109]:
```
rf_random.best_params_
```

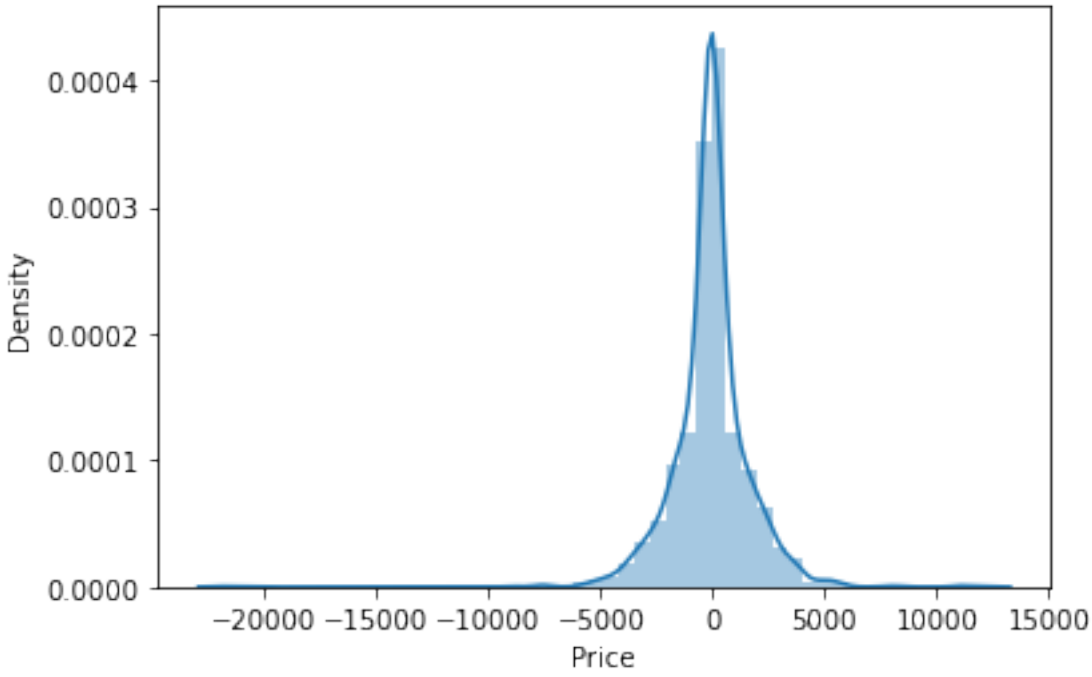
[109]:
```
{'n_estimators': 980,
 'min_samples_split': 14,
 'max_features': 'auto',
 'max_depth': 30}
```

[110]:
```
prediction = rf_random.predict(X_test)
```

[111]:
```
sns.distplot(y_test-prediction)
```

```
C:\Users\khale\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[111]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



[112]:
```
metrics.r2_score(y_test,prediction)
```

[112]:
```
0.8392718547112263
```

[113]:
```
import pickle
```

[114]:
```
filename = 'RandoForestRegressor.sav'
pickle.dump(rf_random, open(filename, 'wb'))
```

[115]:
```
loaded_model = pickle.load(open(filename, 'rb'))
```

[116]:
```
loaded_model
```

[116]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [5, 13, 21, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_split': [5, 10, 14, 100],
                                        'n_estimators': [100, 320, 540, 760,
                                                         980, 1200]},
                   verbose=2)
```

[117]:
```
predictions2=loaded_model.predict(X_test)
```

[118]:
```
metrics.r2_score(y_test,predictions2)
```

[118]: 0.8392718547112263

[ ]: