HOW Many bookings were cancelled?

Ans - [18:31, 3/22/2022] Jahin ML Club: import random
import datetime

```
# Global List Declaration
name = []
phno = []
add = []
checkin = []
checkout = []
room = []
price = []
rc = []
p = []
roomno = []
custid = []
day = []

# Global Variable Declaration

i = 0

# Home Function
def Home():

        print("\t\t\t\t\t\t WELCOME TO HOTEL ANCASA\n")
        print("\t\t\t 1 Booking\n")
        print("\t\t\t 2 Rooms Info\n")
        print("\t\t\t 3 Room Service(Menu Card)\n")
        print("\t\t\t 4 Payment\n")
        print("\t\t\t 5 Record\n")
        print("\t\t\t 0 Exit\n")

        ch=int(input("->"))

        if ch == 1:
                print(" ")
                Booking()

        elif ch == 2:
                print(" ")
                Rooms_Info()

        elif ch == 3:
                print(" ")
                restaurant()

        elif ch == 4:
                print(" ")
```

```
            Payment()

        elif ch == 5:
            print(" ")
            Record()

        else:
            exit()

# Function used in booking

def date(c):

        if c[2] >= 2019 and c[2] <= 2020:

            if c[1] != 0 and c[1] <= 12:

                if c[1] == 2 and c[0] != 0 and c[0] <= 31:

                    if c[2]%4 == 0 and c[0] <= 29:
                        pass

                    elif c[0]<29:
                        pass

                    else:
                        print("Invalid date\n")
                        name.pop(i)
                        phno.pop(i)
                        add.pop(i)
                        checkin.pop(i)
                        checkout.pop(i)
                        Booking()


                # if month is odd & less than equal
                # to 7th month
                elif c[1] <= 7 and c[1]%2 != 0 and c[0] <= 31:
                    pass

                # if month is even & less than equal to 7th
                # month and not 2nd month
                elif c[1] <= 7 and c[1]%2 == 0 and c[0] <= 30 and c[1] != 2:
                    pass

                # if month is even & greater than equal
                # to 8th month
                elif c[1] >= 8 and c[1]%2 == 0 and c[0] <= 31:
                    pass

                # if month is odd & greater than equal
```

```
                    # to 8th month
                    elif c[1]>=8 and c[1]%2!=0 and c[0]<=30:
                            pass

                    else:
                            print("Invalid date\n")
                            name.pop(i)
                            phno.pop(i)
                            add.pop(i)
                            checkin.pop(i)
                            checkout.pop(i)
                            Booking()

            else:
                    print("Invalid date\n")
                    name.pop(i)
                    phno.pop(i)
                    add.pop(i)
                    checkin.pop(i)
                    checkout.pop(i)
                    Booking()

    else:
            print("Invalid date\n")
            name.pop(i)
            phno.pop(i)
            add.pop(i)
            checkin.pop(i)
            checkout.pop(i)
            Booking()


# Booking function
def Booking():

            # used global keyword to
            # use global variable 'i'
            global i
            print(" BOOKING ROOMS")
            print(" ")

            while 1:
                    n = str(input("Name: "))
                    p1 = str(input("Phone No.: "))
                    a = str(input("Address: "))

                    # checks if any field is not empty
                    if n!="" and p1!="" and a!="":
                            name.append(n)
                            add.append(a)
                            break
```

```
        else:
                print("\tName, Phone no. & Address cannot be empty..!!")

cii=str(input("Check-In: "))
checkin.append(cii)
cii=cii.split('/')
ci=cii
ci[0]=int(ci[0])
ci[1]=int(ci[1])
ci[2]=int(ci[2])
date(ci)

coo=str(input("Check-Out: "))
checkout.append(coo)
coo=coo.split('/')
co=coo
co[0]=int(co[0])
co[1]=int(co[1])
co[2]=int(co[2])

# checks if check-out date falls after
# check-in date
if co[1]<ci[1] and co[2]<ci[2]:

        print("\n\tErr..!!\n\tCheck-Out date must fall after Check-In\n")
        name.pop(i)
        add.pop(i)
        checkin.pop(i)
        checkout.pop(i)
        Booking()
elif co[1]==ci[1] and co[2]>=ci[2] and co[0]<=ci[0]:

        print("\n\tErr..!!\n\tCheck-Out date must fall after Check-In\n")
        name.pop(i)
        add.pop(i)
        checkin.pop(i)
        checkout.pop(i)
        Booking()
else:
        pass

date(co)
d1 = datetime.datetime(ci[2],ci[1],ci[0])
d2 = datetime.datetime(co[2],co[1],co[0])
d = (d2-d1).days
day.append(d)

print("----SELECT ROOM TYPE----")
print(" 1. Standard Non-AC")
print(" 2. Standard AC")
```

```
print(" 3. 3-Bed Non-AC")
print(" 4. 3-Bed AC")
print(("\t\tPress 0 for Room Prices"))

ch=int(input("->"))

# if-conditions to display alloted room
# type and it's price
if ch==0:
        print(" 1. Standard Non-AC - Rs. 3500")
        print(" 2. Standard AC - Rs. 4000")
        print(" 3. 3-Bed Non-AC - Rs. 4500")
        print(" 4. 3-Bed AC - Rs. 5000")
        ch=int(input("->"))
if ch==1:
        room.append('Standard Non-AC')
        print("Room Type- Standard Non-AC")
        price.append(3500)
        print("Price- 3500")
elif ch==2:
        room.append('Standard AC')
        print("Room Type- Standard AC")
        price.append(4000)
        print("Price- 4000")
elif ch==3:
        room.append('3-Bed Non-AC')
        print("Room Type- 3-Bed Non-AC")
        price.append(4500)
        print("Price- 4500")
elif ch==4:
        room.append('3-Bed AC')
        print("Room Type- 3-Bed AC")
        price.append(5000)
        print("Price- 5000")
else:
        print(" Wrong choice..!!")


# randomly generating room no. and customer
# id for customer
rn = random.randrange(40)+300
cid = random.randrange(40)+10


# checks if alloted room no. & customer
# id already not alloted
while rn in roomno or cid in custid:
        rn = random.randrange(60)+300
        cid = random.randrange(60)+10

rc.append(0)
```

```
                p.append(0)

                if p1 not in phno:
                        phno.append(p1)
                elif p1 in phno:
                        for n in range(0,i):
                                if p1== phno[n]:
                                        if p[n]==1:
                                                phno.append(p1)
                elif p1 in phno:
                        for n in range(0,i):
                                if p1== phno[n]:
                                        if p[n]==0:
                                                print("\tPhone no. already exists and payment yet
not done..!!")
                                                name.pop(i)
                                                add.pop(i)
                                                checkin.pop(i)
                                                checkout.pop(i)
                                                Booking()
                print("")
                print("\t\t\t**ROOM BOOKED SUCCESSFULLY**\n")
                print("Room No. - ",rn)
                print("Customer Id - ",cid)
                roomno.append(rn)
                custid.append(cid)
                i=i+1
                n=int(input("0-BACK\n ->"))
                if n==0:
                        Home()
                else:
                        exit()

# ROOMS INFO
def Rooms_Info():
        print("              ------ HOTEL ROOMS INFO ------")
        print("")
        print("STANDARD NON-AC")
        print("---------------------------------------------------------------")
        print("Room amenities include: 1 Double Bed, Television, Telephone,")
        print("Double-Door Cupboard, 1 Coffee table with 2 sofa, Balcony and")
        print("an attached washroom with hot/cold water.\n")
        print("STANDARD NON-AC")
        print("---------------------------------------------------------------")
        print("Room amenities include: 1 Double Bed, Television, Telephone,")
        print("Double-Door Cupboard, 1 Coffee table with 2 sofa, Balcony and")
        print("an attached washroom with hot/cold water + Window/Split AC.\n")
        print("3-Bed NON-AC")
        print("---------------------------------------------------------------")
        print("Room amenities include: 1 Double Bed + 1 Single Bed, Television,")
        print("Telephone, a Triple-Door Cupboard, 1 Coffee table with 2 sofa, 1")
```

```
        print("Side table, Balcony with an Accent table with 2 Chair and an")
        print("attached washroom with hot/cold water.\n")
        print("3-Bed AC")
        print("-------------------------------------------------------------")
        print("Room amenities include: 1 Double Bed + 1 Single Bed, Television,")
        print("Telephone, a Triple-Door Cupboard, 1 Coffee table with 2 sofa, ")
        print("1 Side table, Balcony with an Accent table with 2 Chair and an")
        print("attached washroom with hot/cold water + Window/Split AC.\n\n")
        print()
        n=int(input("0-BACK\n ->"))
        if n==0:
                Home()
        else:
                exit()

# RESTAURANT FUNCTION
def restaurant():
        ph=int(input("Customer Id: "))
        global i
        f=0
        r=0
        for n in range(0,i):
                if custid[n]==ph and p[n]==0:
                        f=1
                        print("-------------------------------------------------------------------")
                        print("                                Hotel AnCasa")
                        print("-------------------------------------------------------------------")
                        print("                                    Menu Card")
                        print("-------------------------------------------------------------------")
                        print("\n BEVARAGES                                                    26
Dal Fry..... 140.00")
                        print("--------------------------------- 27 Dal Makhani..... 150.00")
                        print(" 1 Regular Tea..... 20.00        28 Dal Tadka..... 150.00")
                        print(" 2 Masala Tea..... 25.00")
                        print(" 3 Coffee..... 25.00             ROTI")
                        print(" 4 Cold Drink..... 25.00         ----------------------------------")
                        print(" 5 Bread Butter..... 30.00       29 Plain Roti..... 15.00")
                        print(" 6 Bread Jam..... 30.00          30 Butter Roti..... 15.00")
                        print(" 7 Veg. Sandwich..... 50.00      31 Tandoori Roti..... 20.00")
                        print(" 8 Veg. Toast Sandwich..... 50.00 32 Butter Naan..... 20.00")
                        print(" 9 Cheese Toast Sandwich... 70.00")
                        print(" 10 Grilled Sandwich..... 70.00  RICE")
                        print("                                                              ----
------------------------------")
                        print(" SOUPS                                                      33
Plain Rice..... 90.00")
                        print("--------------------------------- 34 Jeera Rice..... 90.00")
                        print(" 11 Tomato Soup..... 110.00      35 Veg Pulao..... 110.00")
                        print(" 12 Hot & Sour..... 110.00       36 Peas Pulao..... 110.00")
                        print(" 13 Veg. Noodle Soup..... 110.00")
                        print(" 14 Sweet Corn..... 110.00       SOUTH INDIAN")
```

```
                print(" 15 Veg. Munchow..... 110.00  ----------------------------------")
                print("                                                                37
Plain Dosa..... 100.00")
                print(" MAIN COURSE                                                    38 Onion
Dosa..... 110.00")
                print("---------------------------------- 39 Masala Dosa..... 130.00")
                print(" 16 Shahi Paneer..... 110.00     40 Paneer Dosa..... 130.00")
                print(" 17 Kadai Paneer..... 110.00     41 Rice Idli..... 130.00")
                print(" 18 Handi Paneer..... 120.00     42 Sambhar Vada..... 140.00")
                print(" 19 Palak Paneer..... 120.00")
                print(" 20 Chilli Paneer..... 140.00       ICE CREAM")
                print(" 21 Matar Mushroom..... 140.00          ----------------------------------")
                print(" 22 Mix Veg..... 140.00       43 Vanilla..... 60.00")
                print(" 23 Jeera Aloo..... 140.00       44 Strawberry..... 60.00")
                print(" 24 Malai Kofta..... 140.00       45 Pineapple..... 60.00")
                print(" 25 Aloo Matar..... 140.00       46 Butter Scotch..... 60.00")
                print("Press 0 -to end ")
                ch=1
                while(ch!=0):

                        ch=int(input(" -> "))

                        # if-elif-conditions to assign item
                        # prices listed in menu card
                        if ch==1 or ch==31 or ch==32:
                                rs=20
                                r=r+rs
                        elif ch<=4 and ch>=2:
                                rs=25
                                r=r+rs
                        elif ch<=6 and ch>=5:
                                rs=30
                                r=r+rs
                        elif ch<=8 and ch>=7:
                                rs=50
                                r=r+rs
                        elif ch<=10 and ch>=9:
                                rs=70
                                r=r+rs
                        elif (ch<=17 and ch>=11) or ch==35 or ch==36 or ch==38:
                                rs=110
                                r=r+rs
                        elif ch<=19 and ch>=18:
                                rs=120
                                r=r+rs
                        elif (ch<=26 and ch>=20) or ch==42:
                                rs=140
                                r=r+rs
                        elif ch<=28 and ch>=27:
                                rs=150
                                r=r+rs
```

```
                elif ch<=30 and ch>=29:
                        rs=15
                        r=r+rs
                elif ch==33 or ch==34:
                        rs=90
                        r=r+rs
                elif ch==37:
                        rs=100
                        r=r+rs
                elif ch<=41 and ch>=39:
                        rs=130
                        r=r+rs
                elif ch<=46 and ch>=43:
                        rs=60
                        r=r+rs
                elif ch==0:
                        pass
                else:
                        print("Wrong Choice..!!")
            print("Total Bill: ",r)

            # updates restaurant charges and then
            # appends in 'rc' list
            r=r+rc.pop(n)
            rc.append(r)
        else:
            pass
    if f == 0:
        print("Invalid Customer Id")
    n=int(input("0-BACK\n ->"))
    if n==0:
        Home()
    else:
        exit()


# PAYMENT FUNCTION
def Payment():

    ph=str(input("Phone Number: "))
    global i
    f=0

    for n in range(0,i):
        if ph==phno[n] :

            # checks if payment is
            # not already done
            if p[n]==0:
                f=1
                print(" Payment")
```

```
                print(" --------------------------------")
                print(" MODE OF PAYMENT")

                print(" 1- Credit/Debit Card")
                print(" 2- Paytm/PhonePe")
                print(" 3- Using UPI")
                print(" 4- Cash")
                x=int(input("-> "))
                print("\n Amount: ",(price[n]*day[n])+rc[n])
                print("\n                    Pay For AnCasa")
                print(" (y/n)")
                ch=str(input("->"))

                if ch=='y' or ch=='Y':
                        print("\n\n --------------------------------")
                        print("              Hotel AnCasa")
                        print(" --------------------------------")
                        print("                        Bill")
                        print(" --------------------------------")
                        print(" Name: ",name[n],"\t\n Phone No.: ",phno[n],"\t\n
Address: ",add[n],"\t")

                        print("\n Check-In: ",checkin[n],"\t\n Check-Out:
",checkout[n],"\t")

                        print("\n Room Type: ",room[n],"\t\n Room Charges:
",price[n]*day[n],"\t")

                        print(" Restaurant Charges: \t",rc[n])
                        print(" --------------------------------")
                        print("\n Total Amount: ",(price[n]*day[n])+rc[n],"\t")
                        print(" --------------------------------")
                        print("              Thank You")
                        print("              Visit Again :)")
                        print(" --------------------------------\n")
                        p.pop(n)
                        p.insert(n,1)

                        # pops room no. and customer id from list and
                        # later assigns zero at same position
                        roomno.pop(n)
                        custid.pop(n)
                        roomno.insert(n,0)
                        custid.insert(n,0)

        else:

                for j in range(n+1,i):
                        if ph==phno[j] :
                                if p[j]==0:
                                        pass

                                else:
                                        f=1
```

```
                                                        print("\n\tPayment has been Made :)\n\n")
        if f==0:
                print("Invalid Customer Id")

        n = int(input("0-BACK\n ->"))
        if n == 0:
                Home()
        else:
                exit()

# RECORD FUNCTION
def Record():

        # checks if any record exists or not
        if phno!=[]:
                print("    * HOTEL RECORD *\n")
                print("| Name    | Phone No. | Address   | Check-In | Check-Out  | Room Type     |
Price      |")
                print("-----------------------------------------------------------------------------------------------------
---------------")

                for n in range(0,i):
                        print("|",name[n],"\t
|",phno[n],"\t|",add[n],"\t|",checkin[n],"\t|",checkout[n],"\t|",room[n],"\t|",price[n])

                print("-----------------------------------------------------------------------------------------------------
---------------")

        else:
                print("No Records Found")
        n = int(input("0-BACK\n ->"))
        if n == 0:
                Home()

        else:
                exit()

# Driver Code
Home()
[18:39, 3/22/2022] Jahin ML Club: 4.
    tmpTime = tmpTime.replace(day=1)
    months.append(tmpTime)
    lastMonth = tmpTime.month
```

3.WHAT IS THE PERCENTAGE OF BOOKING EACH YEAR?

ANS-

```
import requests

def getpricetobook(stock):
  BS = requests.get(f"BOOKING.CSV)
  BS = BS.json()
```

```
    print(BS)

getpricetobook('AAPL')
```

4.which is busiest month in hotel?

Ans - dateRange = [datetime.strptime(dateRanges[0], "%Y-%m-%d"),
datetime.strptime(dateRanges[1], "%Y-%m-%d")]

```
months = []

tmpTime = dateRange[0]
oneWeek = timedelta(weeks=1)
tmpTime = tmpTime.replace(day=1)
dateRange[0] = tmpTime
dateRange[1] = dateRange[1].replace(day=1)
lastMonth = tmpTime.month
months.append(tmpTime)
while tmpTime < dateRange[1]:
    if lastMonth != 12:
        while tmpTime.month <= lastMonth:
            tmpTime += oneWeek
        tmpTime = tmpTime.replace(day=1)
        months.append(tmpTime)
        lastMonth = tmpTime.month

    else:
        while tmpTime.month >= lastMonth:
            tmpTime += oneWeek
        tmpTime = tmpTime.replace(day=1)
        months.append(tmpTime)
        lastMonth = tmpTime.month
```

5. FROM WHICH COUNTRY MOST GUSETS COME?

ANS-

```
    def findCountry (country_name):
    try:
        return pycountry.countries.get(name=country_name).alpha_2
    except:
        return ("not founded")
df['country_alpha_2'] = df.apply(lambda row: findCountry(row.Country_Region) , axis = 1)
```

6.HOW LONG PEOPLE STAY IN HOTEL?

ANS –

```
        from datetime import date
d0 = date(2017, 8, 18)
d1 = date(2017, 10, 26)
```

```
delta = d1 - d0
print(delta.days)
```

7.WHICH QUESTIONS WAS MOST BOOKED ACCOMIDATION TYPE(SINGLE,FAMILY,COUPLE) ?

ANS –

```
    class homeageage:

  def _init_(self,rt='',s=0,p=0,r=0,t=0,a=1000,name='',address='',cindate='',coutdate='',rno=1):

    print ("\n\n**WELCOME TO HOTEl DE SUAREZ**\n")

    self.rt=rt

    self.r=r

    self.t=t

    self.p=p

    self.s=s
    self.a=a
    self.name=name
    self.address=address
    self.cindate=cindate
    self.coutdate=coutdate
    self.rno=rno
  def inputdata(self):
    self.name=input("\nEnter your Fullname:")
    self.address=input("\nEnter your address:")
    self.cindate=input("\nEnter your check in date:")
    self.coutdate=input("\nEnter your checkout date:")
    print("Your room no.:",self.rno,"\n")

  def roomrent(self):#sel1353

    print ("We have the following rooms for you:-")

    print ("1.  Class A----&gt;4000")

    print ("2.  Class B----&gt;3000")

    print ("3.  Class C----&gt;2000")

    print ("4.  Class D----&gt;1000")

    x=int(input("Enter the number of your choice Please-&gt;"))

    n=int(input("For How Many Nights Did You Stay:"))

    if(x==1):
```

```
        print ("you have choose room Class A")

        self.s=4000*n

    elif (x==2):

        print ("you have choose room Class B")

        self.s=3000*n

    elif (x==3):

        print ("you have choose room Class C")

        self.s=2000*n

    elif (x==4):
        print ("you have choose room Class D")

        self.s=1000*n

    else:

        print ("please choose a room")

    print ("your choosen room rent is =",self.s,"\n")

  def foodpurchased(self):

    print("**RESTAURANT MENU**")

    print("1.Dessert-----&gt;100","2.Drinks-----&gt;50","3.Breakfast---&gt;90","4.Lunch----
&gt;110","5.Dinner---&gt;150","6.Exit")


    while (1):

        c=int(input("Enter the number of your choice:"))


        if (c==1):
            d=int(input("Enter the quantity:"))
            self.r=self.r+100*d

        elif (c==2):
             d=int(input("Enter the quantity:"))
             self.r=self.r+50*d

        elif (c==3):
             d=int(input("Enter the quantity:"))
```

```
            self.r=self.r+90*d

        elif (c==4):
            d=int(input("Enter the quantity:"))
            self.r=self.r+110*d

        elif (c==5):
            d=int(input("Enter the quantity:"))
            self.r=self.r+150*d

        elif (c==6):
           break;
        else:
           print("You've Enter an Invalid Key")

    print ("Total food Cost=Rs",self.r,"\n")



  def display(self):
    print ("***HOTEL BILL***")
    print ("Customer details:")
    print ("Customer name:",self.name)
    print ("Customer address:",self.address)
    print ("Check in date:",self.cindate)
    print ("Check out date",self.coutdate)
    print ("Room no.",self.rno)
    print ("Your Room rent is:",self.s)
    print ("Your Food bill is:",self.r)

    self.rt=self.s+self.t+self.p+self.r

    print ("Your sub total Purchased is:",self.rt)
    print ("Additional Service Charges is",self.a)
    print ("Your grandtotal Purchased is:",self.rt+self.a,"\n")
    self.rno+=1




def main():

  a=hotelmanage()


  while (1):
    print("1.Enter Customer Data")
```

```
        print("2.Calculate Room Rent")

        print("3.Calculate Food Purchased")

        print("4.Show total cost")

        print("5.EXIT")

        b=int(input("\nEnter the number of your choice:"))
        if (b==1):
           a.inputdata()

        if (b==2):

           a.roomrent()

        if (b==3):

           a.foodpurchased()

        if (b==4):

           a.display()

        if (b==5):

           quit()


main()
```


8.PREDICTIVE MODELS TO MAKE PREDICTIONS IN FUTER WHETHER THE BOOKING WILL BE CANCELLED OR NOT?

ANS-
Feature Selection Tools
Three different feature selection tools are used to analyse this dataset:
ExtraTreesClassifier: The purpose of the ExtraTreesClassifier is to fit a number of randomized decision trees to the data, and in this regard is a from of ensemble learning. Particularly, random splits of all observations are carried out to ensure that the model does not overfit the data.
Step forward and backward feature selection: This is a “wrapper-based” feature selection method, where the feature selection is based on a specific machine learning algorithm (in this case, the RandomForestClassifier). For forward-step selection, each individual feature is added to the model one at a time, and the features with the highest ROC_AUC score are selected as the best features. When conducting backward feature selection, this process happens in reverse — whereby each feature is dropped from the model one at a time, i.e. the features with the lowest ROC_AUC scores are dropped from the model.
Background and Data Manipulation

The purpose of using these algorithms is to identify features that best help to predict whether a customer will cancel their hotel booking. This is the dependent variable, where (1 = cancel, 0 = follow through with booking).
The features for analysis are as follows.
Interval

```
leadtime = train_df['LeadTime']
arrivaldateyear = train_df['ArrivalDateYear']
arrivaldateweekno = train_df['ArrivalDateWeekNumber']
arrivaldatedayofmonth = train_df['ArrivalDateDayOfMonth']
staysweekendnights = train_df['StaysInWeekendNights']
staysweeknights = train_df['StaysInWeekNights']
adults = train_df['Adults']
children = train_df['Children']
babies = train_df['Babies']
isrepeatedguest = train_df['IsRepeatedGuest']
previouscancellations = train_df['PreviousCancellations']
previousbookingsnotcanceled = train_df['PreviousBookingsNotCanceled']
bookingchanges = train_df['BookingChanges']
agent = train_df['Agent']
company = train_df['Company']
dayswaitinglist = train_df['DaysInWaitingList']
adr = train_df['ADR']
rcps = train_df['RequiredCarParkingSpaces']
totalsqr = train_df['TotalOfSpecialRequests']
```

Categorical

```
arrivaldatemonth = train_df.ArrivalDateMonth.astype("category").cat.codes
arrivaldatemonthcat=pd.Series(arrivaldatemonth)
mealcat=train_df.Meal.astype("category").cat.codes
mealcat=pd.Series(mealcat)
countrycat=train_df.Country.astype("category").cat.codes
countrycat=pd.Series(countrycat)
marketsegmentcat=train_df.MarketSegment.astype("category").cat.codes
marketsegmentcat=pd.Series(marketsegmentcat)
distributionchannelcat=train_df.DistributionChannel.astype("category").cat.codes
distributionchannelcat=pd.Series(distributionchannelcat)
reservedroomtypecat=train_df.ReservedRoomType.astype("category").cat.codes
reservedroomtypecat=pd.Series(reservedroomtypecat)
assignedroomtypecat=train_df.AssignedRoomType.astype("category").cat.codes
assignedroomtypecat=pd.Series(assignedroomtypecat)
deposittypecat=train_df.DepositType.astype("category").cat.codes
deposittypecat=pd.Series(deposittypecat)
customertypecat=train_df.CustomerType.astype("category").cat.codes
customertypecat=pd.Series(customertypecat)
reservationstatuscat=train_df.ReservationStatus.astype("category").cat.codes
reservationstatuscat=pd.Series(reservationstatuscat)
```

With regard to these features, certain features such as lead time are interval — in other words they can take on a wide range of values and are not necessarily constrained by a particular scale. However, certain variables such as customertype are categorical variables. In this regard, cat.codes is used to identify these variables as categorical and ensure that they are not erroneously ranked in the eventual analysis. As an example, consider the following variable: 1 = apple, 2 = banana, 3 = orange.

This variable is categorical and the numbers have no inherent rank — therefore it is important to specify as such.
In this regard — using customertype as an example, the variable is first converted to categorical and then stored as a pandas Series:

```
customertypecat=train_df.CustomerType.astype("category").cat.codes
customertypecat=pd.Series(customertypecat)
```

The IsCanceled variable is the response variable:

```
IsCanceled = train_df['IsCanceled']
y = IsCanceled
```

Once the features have been loaded into Python, they are then stored as a numpy stack (or a sequence of arrays):

```
x =
np.column_stack((leadtime,arrivaldateyear,arrivaldatemonthcat,arrivaldateweekno,arrivaldatedayofmonth,staysweekendnights,staysweeknights,adults,children,babies,mealcat,countrycat,marketsegmentcat,distributionchannelcat,isrepeatedguest,previouscancellations,previousbookingsnotcanceled,reservedroomtypecat,assignedroomtypecat,bookingchanges,deposittypecat,dayswaitinglist,customertypecat,adr,rcps,totalsqr,reservationstatuscat))
x = sm.add_constant(x, prepend=True)
```

Now that the x and y variables have been defined, the feature selection methods are used to identify which variables have the greatest influence on hotel cancellations.
Specifically, once the relevant features have been identified, the SVM (support vector machines) model is used for classification. The identified features from the three techniques outlined above are fed separately into the model to determine which feature selection tool is doing the best job at identifying the important features — which is assumed to be reflected by a higher AUC score.
ExtraTreesClassifier
The ExtraTreesClassifier is generated:

```
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
model.fit(x, y)
print(model.feature_importances_)
```

Here are the results:

```
[0.00000000e+00 2.41288705e-02 6.54290762e-03 3.56552004e-03
 4.69576062e-03 3.47427522e-03 4.05667428e-03 4.86925873e-03
 2.53797514e-03 2.90658184e-03 3.51521069e-04 2.81228056e-03
 3.98090524e-02 1.76395497e-02 5.72618836e-03 4.67231162e-03
 1.06281516e-02 1.18152913e-03 4.53164843e-03 7.05720850e-03
 4.01953363e-03 4.33681743e-02 5.47423587e-04 1.24294822e-02
 7.31621484e-03 2.21889104e-02 7.26745746e-03 7.51675538e-01]
```

Let's sort this into a data frame and take a look at the top features:

```
ext=pd.DataFrame(model.feature_importances_,columns=["extratrees"])
ext
ext.sort_values(['extratrees'], ascending=True)
```

The top identified features are features 1, 12, 13, 21, 23, 25 (lead time, country of origin, market segment, deposit type, customer type, and required car parking spaces). Note that feature 27 (reservation status) is not valid in this case, since this effectively represents the same thing as the response variable — i.e. whether a customer cancelled or followed through with their booking. In this case, including the feature in the analysis would be erroneous.
Step forward and backward feature selection
As previously described, this feature selection method is based on the RandomForestClassifier. In terms of step forward feature selection, the ROC_AUC score is assessed for each feature as it is

added to the model, i.e. the features with the highest scores are added to the model. For step backward feature selection, the process is reversed — features are dropped from the model based on those with the lowest ROC_AUC scores. The top six features are being selected from the dataset using this feature selection tool.

The forward feature selection is implemented as follows:

```
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import roc_auc_score
from mlxtend.feature_selection import SequentialFeatureSelector
forward_feature_selector = SequentialFeatureSelector(RandomForestClassifier(n_jobs=-1),
        k_features=6,
        forward=True,
        verbose=2,
        scoring='roc_auc',
        cv=4)

fselector = forward_feature_selector.fit(x, y)
```

Here is the generated output:

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:   40.8s finished
[2020-03-01 19:01:14] Features: 1/6 -- score: 1.0[Parallel(n_jobs=1)]: Using backend
SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.3s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:   37.4s finished
[2020-03-01 19:01:52] Features: 2/6 -- score: 1.0[Parallel(n_jobs=1)]: Using backend
SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.5s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:   37.3s finished
...
[2020-03-01 19:03:49] Features: 5/6 -- score: 1.0[Parallel(n_jobs=1)]: Using backend
SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:   40.7s finished
[2020-03-01 19:04:30] Features: 6/6 -- score: 1.0
```

We can identify the feature names (or numbers in this case, as they are stored in the array) as follows:

```
>>> fselector.k_feature_names_
('0', '1', '2', '3', '4', '27')
```

The backward feature selection method is more computationally-intensive, as all features in the dataset are being considered.

We implement this by simply setting forward=False.

```
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import roc_auc_score
backward_feature_selector = SequentialFeatureSelector(RandomForestClassifier(n_jobs=-1),
        k_features=6,
        forward=False,
        verbose=2,
        scoring='roc_auc',
        cv=4)

bselector = backward_feature_selector.fit(x, y)
```