Table of Contents

Section 1: Dataset Preparation

In [1]:

```
import numpy as np
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
path = '/Users/jh/Desktop/5900/data/csv files/'
cap = pd.read_csv(path + 'Capacity Info.csv')
code = pd.read_csv(path + 'Airport codes.csv')
dist = pd.read_csv(path + 'Airport Distances.csv')
ppdew = pd.read_csv(path + 'Pax per day each way.csv')
dds = pd.read_csv(path + 'dds.csv')
```

In [109]:

```
path2 = '/Users/jh/Desktop/5900/data/Produced data/'
pop = pd.read_csv(path2 + 'population_cleaned.csv')
gdp = pd.read_csv(path2 + 'GDP_cleaned.csv')
```

In [111]:

```
pop = pop[['city','Avg']]
gdp = gdp[['city','Avg']]
```

In [4]:

```
## language dataset
# lang = pd.io.stata.read_stata('/Users/gyr/Desktop/5900/data/ling_web.dta')
# lang.to_csv('language.csv')
```

1.1 Change columns name

In [5]:

```
cap.columns = ['Date','Op_Airline_Code', 'Origin_Code','Origin_Country','Origin_Cont
               'Destination_Code','Destination_Country','Destination_Continent',
                'Equipment_Code','Type','Flights','Seats','ASKs','Block_Mins','Total
```

In [6]:

```
code.columns = ['Airport','Metro_Code', 'Name','Country_Name','Country_Code','Subreg
```

In [7]:

```
dist.columns = ['RT','Distance']
```

In [8]:

```
ppdew.columns = ['RT','PPDEW','Actual_Fare','Blended_Fare']
```

In [112]:

```
pop.rename(columns = {'Avg':'Avg_pop'},inplace = True)
```

In [113]:

```
gdp.rename(columns = {'Avg':'Avg_gdp'},inplace = True)
```

1.2 Change date type

In [120]:

```
## Observe dataset
# code.info()
# dist.info()
# cap.info()
# ppdew.info()
# dds.info()
# pop.info()
# gdp.info()
```

In [10]:

```
## Modify the date
cap['Date'] = pd.to_datetime(cap['Date']).dt.date
# cap.head()
```

In [117]:

```
## Modify numeric value
cap['Seats'] = cap['Seats'].str.replace(',','')
cap['ASKs'] = cap['ASKs'].str.replace(',','')
cap['Block_Mins'] = cap['Block_Mins'].str.replace(',','')
cap['Total_Kilometers'] = cap['Total_Kilometers'].str.replace(',','')

pop['Avg_pop'] = pop['Avg_pop'].str.replace(',','')
gdp['Avg_gdp'] = gdp['Avg_gdp'].str.replace(',','')
```

In [118]:

```
cap['Seats'] = cap['Seats'].astype(float)
cap['ASKs'] = cap['ASKs'].astype(float)
cap['Block_Mins'] = cap['Block_Mins'].astype(float)
cap['Total_Kilometers'] = cap['Total_Kilometers'].astype(float)

pop['Avg_pop'] = pop['Avg_pop'].astype(float)
gdp['Avg_gdp'] = gdp['Avg_gdp'].astype(float)
```

In [13]:

```
## Modify the travel date
dds.travel_date.value_counts()
dds['travel_date'] = pd.to_datetime(dds['travel_date']).dt.date
dds.head()
```

Out[13]:

	travel_date	airline	origin	destination	pax	pax_poo_origin	pax_poo_destination	pax_poo_ot
0	2019-03-01	NaN	BEL	BVS	9.0	2.92194	5.84244	0.23
1	2019-05-01	NaN	BEL	BVS	20.0	0.00000	20.00000	0.00
2	2019-03-01	NaN	BEL	MEU	11.0	9.99999	0.00000	1.00
3	2019-05-01	NaN	BEL	MEU	13.0	12.00004	0.00000	0.99
4	2019-06-01	NaN	BEL	MEU	18.0	16.99992	0.00000	1.00

1.3 Merge and split columns

In [14]:

```
## cap
cap['RT'] = cap['Origin_Code'] + cap['Destination_Code']
cap['Distance'] = round(cap['Total_Kilometers']/cap['Flights'],0)
```

In [15]:

```
## dist
dist['Origin_Code'] = dist['RT'].str[0:3]
dist['Destination_Code'] = dist['RT'].str[-3:]
```

In [16]:

```
## ppdew
ppdew['Avg_Fare'] = ppdew[['Actual_Fare','Blended_Fare']].mean(axis=1)
ppdew['Origin_Code'] = ppdew['RT'].str[0:3]
ppdew['Destination_Code'] = ppdew['RT'].str[-3:]
```

In [17]:

```
## dds
dds['RT'] = dds['origin'] + dds['destination']
dds['rev_pax'] = dds['revenue']/dds['pax']
```

1.4 Check for missing values

In [18]:

```
def missing_values(data):
    missing_values = data.isnull().sum()
    missing_values_per = (data.isnull().sum() / data.isnull().count())
    mis = pd.concat([missing_values, missing_values_per],axis=1, keys = ['Count_of_M
    return mis
```

In [19]:

```
missing_values(cap)
```

Out[19]:

	Count_of_Missing	Percentage
Date	0	0.0
Op_Airline_Code	0	0.0
Origin_Code	0	0.0
Origin_Country	0	0.0
Origin_Continent	0	0.0
Destination_Code	0	0.0
Destination_Country	0	0.0
Destination_Continent	0	0.0
Equipment_Code	0	0.0
Type	0	0.0
Flights	0	0.0
Seats	0	0.0
ASKs	0	0.0
Block_Mins	0	0.0
Total_Kilometers	0	0.0
RT	0	0.0
Distance	0	0.0

In [20]:

```
missing_values(code)
```

Out[20]:

	Count_of_Missing	Percentage
Airport	0	0.000000
Metro_Code	0	0.000000
Name	5180	0.519611
Country_Name	0	0.000000
Country_Code	24	0.002407
Subregion	0	0.000000
Region	0	0.000000

The country code of country NAMIBIA is 'NA', so there is no missing values for its country code. I will recode with 'NAM' instead.

In [21]:

```
code[code.Country_Code.isnull()]
code['Country_Code'].fillna('NAM',inplace = True)
```

In [22]:

```
code[code.Name.isnull()]
```

Out[22]:

	Airport	Metro_Code	Name	Country_Name	Country_Code	Subregion	Region
5	AAF	AAF	NaN	UNITED STATES	US	NAM	NAM
9	AAJ	AAJ	NaN	SURINAME	SR	LOSA	SAM
10	AAK	AAK	NaN	KIRIBATI	KI	SWP	SWP
14	AAO	AAO	NaN	VENEZUELA	VE	UPSA	SAM
18	AAS	AAS	NaN	INDONESIA	ID	SEA	SEA
...	...	...	...	...	...	...	...
9963	ZZO	ZZO	NaN	RUSSIAN FEDERATION	RU	EUR	EUR
9964	ZZR	ZZR	NaN	UNITED STATES	US	NAM	NAM
9965	ZZT	ZZT	NaN	UNITED STATES	US	NAM	NAM
9967	ZZV	ZZV	NaN	UNITED STATES	US	NAM	NAM
9968	ZZZ	ZZZ	NaN	CANADA	CA	NAM	NAM

5180 rows × 7 columns

In [23]:

```
missing_values(dist)
```

Out[23]:

	Count_of_Missing	Percentage
RT	0	0.0
Distance	0	0.0
Origin_Code	0	0.0
Destination_Code	0	0.0

In [24]:

```
missing_values(ppdew)
```

Out[24]:

	Count_of_Missing	Percentage
RT	0	0.000000
PPDEW	0	0.000000
Actual_Fare	2464	0.590604
Blended_Fare	0	0.000000
Avg_Fare	0	0.000000
Origin_Code	0	0.000000
Destination_Code	0	0.000000

In [25]:

```
missing_values(dds)
```

Out[25]:

	Count_of_Missing	Percentage
travel_date	0	0.000000
airline	85	0.000039
origin	0	0.000000
destination	0	0.000000
pax	0	0.000000
pax_poo_origin	0	0.000000
pax_poo_destination	0	0.000000
pax_poo_other	0	0.000000
revenue	0	0.000000
RT	0	0.000000
rev_pax	0	0.000000

In [26]:

```
dds[dds.airline.isnull()]
```

Out[26]:

	travel_date	airline	origin	destination	pax	pax_poo_origin	pax_poo_destination	pax_p
0	2019-03-01	NaN	BEL	BVS	9.0	2.92194	5.84244	
1	2019-05-01	NaN	BEL	BVS	20.0	0.00000	20.00000	
2	2019-03-01	NaN	BEL	MEU	11.0	9.99999	0.00000	
3	2019-05-01	NaN	BEL	MEU	13.0	12.00004	0.00000	
4	2019-06-01	NaN	BEL	MEU	18.0	16.99992	0.00000	
...	...	...	...	...	...	...	...	
883786	2020-03-01	NaN	FOR	QBX	2.0	2.00000	0.00000	
883787	2020-02-01	NaN	FOR	QIG	11.0	9.99999	1.00001	
883788	2020-03-01	NaN	FOR	QIG	6.0	3.00000	3.00000	
883789	2020-01-01	NaN	GRU	QGA	11.0	4.00004	5.99995	
883790	2020-01-01	NaN	RRJ	QJR	2.0	0.37362	1.60440	

85 rows × 11 columns

1.6 Market Segment

dist

In [27]:

```
#Match country names to existing routes
ori_rt = pd.merge(dist,code,left_on=['Origin_Code'],right_on = 'Airport')
df1 = ori_rt.iloc[:,[0,1,2,5,6,7]]

des_rt = pd.merge(dist,code,left_on=['Destination_Code'],right_on = 'Airport')
df2 = des_rt.iloc[:,[3,5,6,7]]

dist_CountryName = pd.concat([df1,df2],axis=1)
dist_CountryName
dist_CountryName.columns = ['RT','Distance','Origin_Code','Origin_Metro_Code','Origi
                            'Destination_Code','Destination_Metro_Code','Destination_
dist_CountryName
```

Out[27]:

	RT	Distance	Origin_Code	Origin_Metro_Code	Origin_City	Origin_Country	Destinat
0	AAZGUA	109.0	AAZ	AAZ	Ambriz	GUATEMALA	
1	ACABOG	3098.0	ACA	ACA	Acapulco	MEXICO	
2	ACAHAV	1940.0	ACA	ACA	Acapulco	MEXICO	
3	ACALIM	4054.0	ACA	ACA	Acapulco	MEXICO	
4	ACAMEX	307.0	ACA	ACA	Acapulco	MEXICO	
...	...	...	...	...	...	...	
6721	XQPYYZ	3832.0	XQP	XQP	Quepos	COSTA RICA	
6722	YHZYUL	804.0	YHZ	YHZ	Halifax	CANADA	
6723	YOWYYZ	363.0	YOW	YOW	Ottawa	CANADA	
6724	YTOYUL	507.0	YTO	YTO	Toronto	CANADA	
6725	YULYYZ	507.0	YUL	YMQ	Montreal	CANADA	

6726 rows × 10 columns

In [28]:

```
# ## add column to divide into domestic flights 国内航班
# dist_CountryName['Domestic'] = np.where((dist_CountryName['Origin_Country'] == 'CC
#                                         (dist_CountryName['Destination_Country'] =
#                                         , 1, 0)
# dist_CountryName
```

Out[28]:

	RT	Distance	Origin_Code	Origin_Metro_Code	Origin_City	Origin_Country	Destinat
0	AAZGUA	109.0	AAZ	AAZ	Ambriz	GUATEMALA	
1	ACABOG	3098.0	ACA	ACA	Acapulco	MEXICO	
2	ACAHAV	1940.0	ACA	ACA	Acapulco	MEXICO	
3	ACALIM	4054.0	ACA	ACA	Acapulco	MEXICO	
4	ACAMEX	307.0	ACA	ACA	Acapulco	MEXICO	
...	...	...	...	...	...	...	
6721	XQPYYZ	3832.0	XQP	XQP	Quepos	COSTA RICA	
6722	YHZYUL	804.0	YHZ	YHZ	Halifax	CANADA	
6723	YOWYYZ	363.0	YOW	YOW	Ottawa	CANADA	
6724	YTOYUL	507.0	YTO	YTO	Toronto	CANADA	
6725	YULYYZ	507.0	YUL	YMQ	Montreal	CANADA	

6726 rows × 11 columns

In [29]:

```
missing_values(dist_CountryName)
```

Out[29]:

	Count_of_Missing	Percentage
RT	0	0.000000
Distance	0	0.000000
Origin_Code	0	0.000000
Origin_Metro_Code	0	0.000000
Origin_City	19	0.002825
Origin_Country	0	0.000000
Destination_Code	0	0.000000
Destination_Metro_Code	0	0.000000
Destination_City	43	0.006393
Destination_Country	0	0.000000
Domestic	0	0.000000

In [30]:

```
#whether to fly across country 跨境
# dist_CountryName['cross_country'] = np.where(dist_CountryName['Origin_Country']==
#                                              dist_CountryName['Destination_Country
#                                       'same','different')
# dist_CountryName

# dist_CountryName.cross_country.value_counts()
```

dds

In [31]:

```
#Match city and country names to routes
dds1 = pd.merge(dds,code[['Airport','Metro_Code','Name','Country_Name']],
                how = 'left',left_on=['origin'],right_on = 'Airport')
dds2 = pd.merge(dds1,code[['Airport','Metro_Code','Name','Country_Name']],
                how = 'left',left_on=['destination'],right_on = 'Airport')

dds_ccName = dds2.iloc[:,[0,1,9,
                          2,13,14,
                          3,17,18,
                          4,8,10]]
dds_ccName.columns = ['Date','Airline','RT','Origin','Origin_City','Origin_Country',
                      'Destination','Destination_City','Destination_Country',
                      'pax','revenue','rev_pax']

dds_ccName
```

Out[31]:

	Date	Airline	RT	Origin	Origin_City	Origin_Country	Destination	Destination_C
0	2019-03-01	NaN	BELBVS	BEL	Belem	BRAZIL	BVS	Na
1	2019-05-01	NaN	BELBVS	BEL	Belem	BRAZIL	BVS	Na
2	2019-03-01	NaN	BELMEU	BEL	Belem	BRAZIL	MEU	Monte Doura
3	2019-05-01	NaN	BELMEU	BEL	Belem	BRAZIL	MEU	Monte Doura
4	2019-06-01	NaN	BELMEU	BEL	Belem	BRAZIL	MEU	Monte Doura
...	...	...	...	...	...	...	...	
2162530	2022-06-01	ZP	ASUEZE	ASU	Asuncion	PARAGUAY	EZE	Buenos Air
2162531	2022-06-01	ZP	ASUMVD	ASU	Asuncion	PARAGUAY	MVD	Montevid
2162532	2022-06-01	ZP	ASUVVI	ASU	Asuncion	PARAGUAY	VVI	Santa Cr
2162533	2022-06-01	ZP	EZEVVI	EZE	Buenos Aires	ARGENTINA	VVI	Santa Cr
2162534	2022-06-01	ZP	MVDVVI	MVD	Montevideo	URUGUAY	VVI	Santa Cr

2162535 rows × 12 columns

In [32]:

```
## add column to divide into domestic flights 国内航班
dds_ccName['Domestic'] = np.where((dds_ccName['Origin_Country'] == 'COLOMBIA') &
                                  (dds_ccName['Destination_Country'] == 'COLOM
                                  , 1, 0)
dds_ccName
```

Out[32]:

	Date	Airline	RT	Origin	Origin_City	Origin_Country	Destination	Destination_C
0	2019-03-01	NaN	BELBVS	BEL	Belem	BRAZIL	BVS	Na
1	2019-05-01	NaN	BELBVS	BEL	Belem	BRAZIL	BVS	Na
2	2019-03-01	NaN	BELMEU	BEL	Belem	BRAZIL	MEU	Monte Doura
3	2019-05-01	NaN	BELMEU	BEL	Belem	BRAZIL	MEU	Monte Doura
4	2019-06-01	NaN	BELMEU	BEL	Belem	BRAZIL	MEU	Monte Doura
...	...	...	...	...	...	...	...	
2162530	2022-06-01	ZP	ASUEZE	ASU	Asuncion	PARAGUAY	EZE	Buenos Air
2162531	2022-06-01	ZP	ASUMVD	ASU	Asuncion	PARAGUAY	MVD	Montevid
2162532	2022-06-01	ZP	ASUVVI	ASU	Asuncion	PARAGUAY	VVI	Santa Cr
2162533	2022-06-01	ZP	EZEVVI	EZE	Buenos Aires	ARGENTINA	VVI	Santa Cr
2162534	2022-06-01	ZP	MVDVVI	MVD	Montevideo	URUGUAY	VVI	Santa Cr

2162535 rows × 13 columns

In [33]:

```
missing_values(dds_ccName)
```

Out[33]:

	Count_of_Missing	Percentage
Date	0	0.000000
Airline	85	0.000039
RT	0	0.000000
Origin	0	0.000000
Origin_City	17262	0.007982
Origin_Country	308	0.000142
Destination	0	0.000000
Destination_City	16863	0.007798
Destination_Country	1351	0.000625
pax	0	0.000000
revenue	0	0.000000
rev_pax	0	0.000000
Domestic	0	0.000000

比对

In [34]:

```
def same_element(list1,list2):
    set1 = set(list1)
    set2 = set(list2)
    return (set1 & set2),(set1 ^ set2),((set1|set2)-set2),((set1|set2)-set1)
```

In [80]:

```
same,dif,alone_forward,alone_backward = same_element(dds_dp_reverse['RT'],dds_dp['RT
print('相同元素个数：',len(same))
print('不相同的元素个数：',len(dif))
print('列表1有列表2没有的元素个数：',len(alone_forward))
print('列表1没有列表2有的元素个数：',len(alone_backward))
```

相同元素个数： 0
不相同的元素个数： 3547
列表1有列表2没有的元素个数： 1665
列表1没有列表2有的元素个数： 1882

In [95]:

```
same,dif,alone_forward,alone_backward = same_element(dds_all['Origin_City'],pop['cit
print('相同元素个数：',len(same))
print('不相同的元素个数：',len(dif))
print('列表1有列表2没有的元素个数：',len(alone_forward))
print('列表1没有列表2有的元素个数：',len(alone_backward))
```

相同元素个数： 21
不相同的元素个数： 851
列表1有列表2没有的元素个数： 184
列表1没有列表2有的元素个数： 667

In [ ]:

```

```

Section 2: Merge Data

cap & dist

In [197]:

```
capDist = cap[['RT','Distance','Origin_Code','Destination_Code']]
capDist = capDist.drop_duplicates(['RT'])
```

In [198]:

```
all_dist = pd.concat([capDist,dist],axis=0).drop_duplicates(['RT']).reset_index(drop
all_dist
```

Out[198]:

	RT	Distance	Origin_Code	Destination_Code
0	AGPOTP	2715.0	AGP	OTP
1	AMSOTP	1788.0	AMS	OTP
2	BCMDUB	2467.0	BCM	DUB
3	BCMLTN	2055.0	BCM	LTN
4	CLJDUB	2246.0	CLJ	DUB
...	...	...	...	...
44625	VVIYTO	7024.0	VVI	YTO
44626	VVIYUL	7101.0	VVI	YUL
44627	VVIYYZ	7025.0	VVI	YYZ
44628	XQPYYZ	3832.0	XQP	YYZ
44629	YTOYUL	507.0	YTO	YUL

44630 rows × 4 columns

all_dist & ppdew

In [208]:

```
## 正交
dist_ppdew = pd.merge(all_dist,
                      ppdew,
                      how = 'inner',
                      left_on=['RT','Origin_Code','Destination_Code'],
                      right_on = ['RT','Origin_Code','Destination_Code'])

dist_ppdew = dist_ppdew.iloc[:,[0,1,2,3,4,7]]
dist_ppdew
```

Out[208]:

	RT	Distance	Origin_Code	Destination_Code	PPDEW	Avg_Fare
0	GYEJFK	4778.0	GYE	JFK	479.432877	235.568545
1	AIRJIA	158.0	AIR	JIA	3.049315	117.211590
2	BELSTM	703.0	BEL	STM	227.527397	103.164672
3	CIZMAO	362.0	CIZ	MAO	7.915068	70.708394
4	CNFIOS	748.0	CNF	IOS	82.376712	126.527077
...	...	...	...	...	...	...
2184	TPPTRU	349.0	TPP	TRU	12.367123	113.900917
2185	UIOVVI	2571.0	UIO	VVI	3.624658	322.852233
2186	UIOYUL	5095.0	UIO	YUL	5.501370	353.822060
2187	UIOYYZ	4875.0	UIO	YYZ	18.563014	277.677592
2188	VVIYYZ	7025.0	VVI	YYZ	3.068493	457.038579

2189 rows × 6 columns

In [210]:

```
## 反交
dist_ppdew_reverse = pd.merge(all_dist,
                              ppdew,
                              how = 'inner',
                              left_on=['Origin_Code','Destination_Code'],
                              right_on = ['Destination_Code','Origin_Code'])

dist_ppdew_reverse = dist_ppdew_reverse.iloc[:,[0,1,2,3,5,8]]
dist_ppdew_reverse.columns = ['RT','Distance','Origin_Code','Destination_Code',
                              'PPDEW','Avg_Fare']
dist_ppdew_reverse
```

Out[210]:

	RT	Distance	Origin_Code	Destination_Code	PPDEW	Avg_Fare
0	JFKGYE	4778.0	JFK	GYE	479.432877	235.568545
1	IOSCNF	748.0	IOS	CNF	82.376712	126.527077
2	JIAAIR	158.0	JIA	AIR	3.049315	117.211590
3	MAOCIZ	362.0	MAO	CIZ	7.915068	70.708394
4	POACWB	533.0	POA	CWB	338.954794	75.396635
...	...	...	...	...	...	...
1353	YYCBZE	4365.0	YYC	BZE	5.272603	246.274913
1354	MCZAEP	3607.0	MCZ	AEP	12.708493	199.758537
1355	NATAEP	3972.0	NAT	AEP	23.061519	294.789778
1356	RECAEP	3787.0	REC	AEP	27.440100	286.570249
1357	ROSGIG	2054.0	ROS	GIG	6.924658	279.237250

1358 rows × 6 columns

In [211]:

```
dp = pd.concat([dist_ppdew,dist_ppdew_reverse],axis=0).drop_duplicates(['RT']).reset
dp
```

Out[211]:

	RT	Distance	Origin_Code	Destination_Code	PPDEW	Avg_Fare
0	GYEJFK	4778.0	GYE	JFK	479.432877	235.568545
1	AIRJIA	158.0	AIR	JIA	3.049315	117.211590
2	BELSTM	703.0	BEL	STM	227.527397	103.164672
3	CIZMAO	362.0	CIZ	MAO	7.915068	70.708394
4	CNFIOS	748.0	CNF	IOS	82.376712	126.527077
...	...	...	...	...	...	...
3542	YYCBZE	4365.0	YYC	BZE	5.272603	246.274913
3543	MCZAEP	3607.0	MCZ	AEP	12.708493	199.758537
3544	NATAEP	3972.0	NAT	AEP	23.061519	294.789778
3545	RECAEP	3787.0	REC	AEP	27.440100	286.570249
3546	ROSGIG	2054.0	ROS	GIG	6.924658	279.237250

3547 rows × 6 columns

dds & dp -> freq_rt

In [52]:

```
### dds does not exit the same route with opposite origin and destination

# dds_ccName[dds_ccName.RT == 'BOGAUA']
# dds_ccName[dds_ccName.RT == 'AUABOG']
```

Out[52]:

	Date	Airline	RT	Origin	Origin_City	Origin_Country	Destination	Destination_C
140451	2019-01-01	AV	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
140452	2019-02-01	AV	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
140453	2019-03-01	AV	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
140454	2019-04-01	AV	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
140455	2019-05-01	AV	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
...	...	...	...	...	...	...	...	
2056541	2022-03-01	P5	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
2094674	2022-06-01	AA	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
2114194	2022-06-01	AV	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
2119450	2022-06-01	CM	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste
2149519	2022-06-01	P5	BOGAUA	BOG	Bogota	COLOMBIA	AUA	Aru (Oranjeste

129 rows × 13 columns

In [212]:

```
## 正交
dds_dp = pd.merge(dp,
                  dds_ccName,
                  how = 'inner',
                  left_on=['Origin_Code','Destination_Code'],
                  right_on = ['Origin','Destination'])

dds_dp = dds_dp.iloc[:,[6,7,8,9,10,11,12,13,14,15,16,17,1,4,5,18]]
dds_dp.rename(columns = {'RT_y':'RT',
                         'Distance':'distance'},
              inplace = True)
```

In [213]:

```
## 反交
dds_dp_reverse = pd.merge(dp,
                          dds_ccName,
                          how = 'inner',
                          left_on=['Origin_Code','Destination_Code'],
                          right_on = ['Destination','Origin'])

dds_dp_reverse = dds_dp_reverse.iloc[:,[6,7,0,9,10,11,12,13,14,15,16,17,1,4,5,18]]
dds_dp_reverse.rename(columns = {'RT_x':'RT',
                                 'Distance':'distance'},
                      inplace = True)
```

In [214]:

```
dds_dp_reverse
```

Out[214]:

	Date	Airline	RT	Origin	Origin_City	Origin_Country	Destination	Destination_Cit
0	2019-01-01	AV	AUABOG	BOG	Bogota	COLOMBIA	AUA	Arub (Oranjestac
1	2019-02-01	AV	AUABOG	BOG	Bogota	COLOMBIA	AUA	Arub (Oranjestac
2	2019-03-01	AV	AUABOG	BOG	Bogota	COLOMBIA	AUA	Arub (Oranjestac
3	2019-04-01	AV	AUABOG	BOG	Bogota	COLOMBIA	AUA	Arub (Oranjestac
4	2019-05-01	AV	AUABOG	BOG	Bogota	COLOMBIA	AUA	Arub (Oranjestac
...	...	...	...	...	...	...	...	.
258615	2022-01-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosari
258616	2022-02-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosari
258617	2022-03-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosari
258618	2022-06-01	AR	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosari
258619	2022-06-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosari

258620 rows × 16 columns

In [221]:

```
dds_dp_all = pd.concat([dds_dp,dds_dp_reverse],axis=0).reset_index(drop=True)
dds_dp_all['less_than_500'] = np.where(dds_dp_all['distance'] <= 500,1,0)
dds_dp_all['fare_per_dist'] = dds_dp_all['Avg_Fare'] / dds_dp_all['distance']
dds_dp_all
```

Out[221]:

	Date	Airline	RT	Origin	Origin_City	Origin_Country	Destination	Destination_City
0	2019-01-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
1	2019-02-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
2	2019-03-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
3	2019-04-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
4	2019-05-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
...	...	...	...	...	...	...	...	...
539294	2022-01-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539295	2022-02-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539296	2022-03-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539297	2022-06-01	AR	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539298	2022-06-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario

539299 rows × 18 columns

In [222]:

```
num = dds_dp_all.groupby('RT')['pax'].count().reset_index()
num.columns = ['RT','Num_Flights']
```

In [223]:

```
dds_all = pd.merge(dds_dp_all,
                   num,
                   on = 'RT')
dds_all
```

Out[223]:

	Date	Airline	RT	Origin	Origin_City	Origin_Country	Destination	Destination_City
0	2019-01-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
1	2019-02-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
2	2019-03-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
3	2019-04-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
4	2019-05-01	AA	GYEJFK	GYE	Guayaquil	ECUADOR	JFK	New York
...	...	...	...	...	...	...	...	...
539294	2022-01-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539295	2022-02-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539296	2022-03-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539297	2022-06-01	AR	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario
539298	2022-06-01	G3	ROSGIG	GIG	Rio de Janeiro	BRAZIL	ROS	Rosario

539299 rows × 19 columns

In [278]:

```
freq_rt = dds_all.groupby('RT')['pax','revenue','rev_pax','Avg_Fare','distance','far
                                'Num_Flights','Domestic','less_than_500'].mean()
freq_rt
```

Out[278]:

RT	pax	revenue	rev_pax	Avg_Fare	distance	fare_per_dist	Num_Fli
ACABOG	31.712500	5376.149438	210.463419	218.726935	3098.0	0.070603	1
ADZASU	129.083333	19099.690952	163.801435	155.635187	4958.0	0.031391	
ADZAXM	313.791045	24684.021940	78.815649	79.111784	1110.0	0.071272	
ADZBAQ	1544.145833	112718.451250	82.049541	79.191370	777.0	0.101919	1
ADZBGA	894.750000	67091.008167	74.847411	75.370439	1118.0	0.067415	1
...	...	...	...	...	...	...	
ZCOBBA	949.900000	44083.028500	74.193615	43.359949	798.0	0.054336	
ZCOPUQ	739.851852	54409.769722	98.494537	85.128228	1601.0	0.053172	1
ZCOSCL	15945.146341	819259.429593	43.091194	54.114815	620.0	0.087282	1
ZOSSCL	5518.742857	289006.037000	51.823934	51.250736	826.0	0.062047	
ZRHGRU	331.397459	198216.328076	535.687619	601.643498	9587.0	0.062756	5

3547 rows × 9 columns

dds_all & gdp, pop -> gdp_pop_rt

In [226]:

```
gdp1  = pd.merge(dds_all,
                 gdp,
                 how = 'inner',
                 left_on = 'Origin_City',
                 right_on = 'city')
gdp2  = pd.merge(gdp1,
                 gdp,
                 how = 'inner',
                 left_on = 'Destination_City',
                 right_on = 'city')

gdp2 = gdp2.drop(['city_x','city_y'],axis = 1)
```

In [227]:

```
pop1  = pd.merge(gdp2,
                 pop,
                 how = 'inner',
                 left_on = 'Origin_City',
                 right_on = 'city')
pop2  = pd.merge(pop1,
                 pop,
                 how = 'inner',
                 left_on = 'Destination_City',
                 right_on = 'city')

gdp_pop = pop2.drop(['city_x','city_y'],axis = 1)
gdp_pop.rename(columns = {'Avg_gdp_x':'Avg_gdp_ori',
                          'Avg_gdp_y':'Avg_gdp_des',
                          'Avg_pop_x':'Avg_pop_ori',
                          'Avg_pop_y':'Avg_pop_des'},
            inplace = True)
gdp_pop
```

Out[227]:

	Date	Airline	RT	Origin	Origin_City	Origin_Country	Destination	Destination_City
0	2019-01-01	ANQ	AXMEOH	AXM	Armenia	COLOMBIA	EOH	Medellir
1	2019-02-01	ANQ	AXMEOH	AXM	Armenia	COLOMBIA	EOH	Medellir
2	2019-03-01	ANQ	AXMEOH	AXM	Armenia	COLOMBIA	EOH	Medellir
3	2019-04-01	ANQ	AXMEOH	AXM	Armenia	COLOMBIA	EOH	Medellir
4	2019-01-01	H1	AXMEOH	AXM	Armenia	COLOMBIA	EOH	Medellir
...	...	...	...	...	...	...	...	..
29216	2022-02-01	LA	PMCCCP	CCP	Concepcion	CHILE	PMC	Puerto Montt
29217	2022-03-01	LA	PMCCCP	CCP	Concepcion	CHILE	PMC	Puerto Montt
29218	2022-06-01	H2	PMCCCP	CCP	Concepcion	CHILE	PMC	Puerto Montt
29219	2022-06-01	JA	PMCCCP	CCP	Concepcion	CHILE	PMC	Puerto Montt
29220	2022-06-01	LA	PMCCCP	CCP	Concepcion	CHILE	PMC	Puerto Montt

29221 rows × 23 columns

In [249]:

```
gdp_pop_rt = gdp_pop.groupby('RT')['Avg_gdp_ori','Avg_gdp_des','Avg_pop_ori','Avg_po
                                   'revenue','pax','rev_pax','Avg_Fare','distance','f
                                   'Num_Flights','Domestic','less_than_500'].mean()
gdp_pop_rt
```

Out[249]:

RT	Avg_gdp_ori	Avg_gdp_des	Avg_pop_ori	Avg_pop_des	revenue	pax	
AMSMDE	13612.9	66348.0	3619208.5	2763163.8	1.886192e+04	52.498630	3
ANFCCP	67887.5	19751.9	368953.0	957097.8	2.921343e+05	4108.569106	
ANFCLO	67887.5	13447.5	368953.0	2528753.3	1.147141e+05	711.660714	1
ATLMDE	13612.9	62391.7	3619208.5	5316074.8	3.048958e+04	154.509434	2
AXMBAQ	8774.4	10685.5	386160.6	2153977.4	2.254852e+04	321.547619	
...	...	...	...	...	...	...	
VVCMDE	13612.9	26477.8	3619208.5	497844.8	2.123372e+04	335.969231	
VVCPEI	10112.5	26477.8	664189.0	497844.8	1.567558e+04	194.623188	
VVILPB	24298.8	48801.1	270037.5	270361.0	2.574870e+06	24567.946518	1
VVIMAD	48801.1	47064.0	270361.0	6711408.8	1.953686e+05	507.624175	3
VVIYYZ	48801.1	44271.2	270361.0	7022976.4	1.641105e+04	45.771084	4

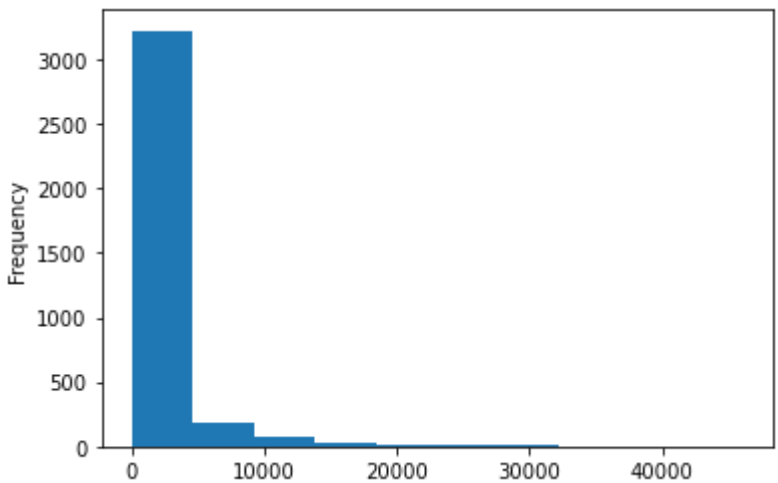
184 rows × 13 columns

Section3: Data Modeling

3.1 Benchmark

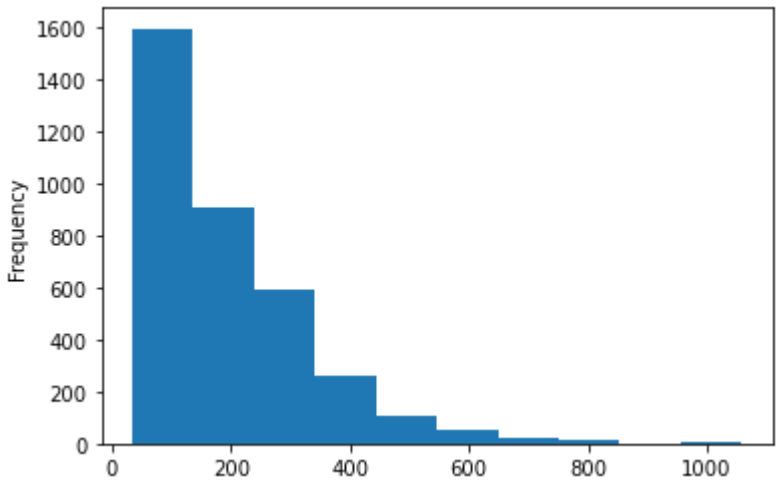
In [317]:

```
freq_rt['pax'].plot(kind='hist')
plt.show()
```



In [318]:

```
freq_rt['rev_pax'].plot(kind='hist')
plt.show()
```



In [343]:

```
freq_rt[['pax','rev_pax']].median()
```

Out[343]:

```
pax         651.817073
rev_pax     147.716361
dtype: float64
```

In [322]:

```
freq_rt[['pax','rev_pax']].mean()
```

Out[322]:

```
pax         1826.584944
rev_pax      193.715730
dtype: float64
```

3.2 Log transformations

In [229]:

```
from __future__ import division
import statsmodels.api as sm
import statsmodels.formula.api as smf
import os
```

In [271]:

```
sub = gdp_pop_rt[['rev_pax', 'revenue','pax',
                  'Avg_gdp_ori','Avg_gdp_des','Avg_pop_ori','Avg_pop_des',
                  'Avg_Fare','distance','fare_per_dist','Num_Flights',
                  'Domestic','less_than_500']]
```

In [273]:

```
sub["ln_rev_pax"] = np.log(sub["rev_pax"])
sub["ln_pax"] = np.log(sub["pax"])

sub["ln_Avg_gdp_ori"] = np.log(sub["Avg_gdp_ori"])
sub["ln_Avg_gdp_des"] = np.log(sub["Avg_gdp_des"])
sub["ln_Avg_pop_ori"] = np.log(sub["Avg_pop_ori"])
sub["ln_Avg_pop_des"] = np.log(sub["Avg_pop_des"])
sub["ln_distance"] = np.log(sub["distance"])
sub["ln_Avg_Fare"] = np.log(sub["Avg_Fare"])
sub["ln_Num_Flights"] = np.log(sub["Num_Flights"])
```

In [274]:

```
sub
```

Out[274]:

RT	rev_pax	revenue	pax	Avg_gdp_ori	Avg_gdp_des	Avg_pop_ori	Avg
AMSMDE	372.470630	1.886192e+04	52.498630	13612.9	66348.0	3619208.5	
ANFCCP	69.893120	2.921343e+05	4108.569106	67887.5	19751.9	368953.0	
ANFCLO	184.539385	1.147141e+05	711.660714	67887.5	13447.5	368953.0	
ATLMDE	238.288747	3.048958e+04	154.509434	13612.9	62391.7	3619208.5	
AXMBAQ	71.894237	2.254852e+04	321.547619	8774.4	10685.5	386160.6	
...	...	...	...	...	...	...	
VVCMDE	70.319397	2.123372e+04	335.969231	13612.9	26477.8	3619208.5	
VVCPEI	75.051173	1.567558e+04	194.623188	10112.5	26477.8	664189.0	
VVILPB	107.181065	2.574870e+06	24567.946518	24298.8	48801.1	270037.5	
VVIMAD	387.755535	1.953686e+05	507.624175	48801.1	47064.0	270361.0	
VVIYYZ	410.616988	1.641105e+04	45.771084	48801.1	44271.2	270361.0	

184 rows × 22 columns

Model 1: Geo-economic

Split Dataset

In [331]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test  = train_test_split(sub.drop('pax',axis=1),
                                                  sub['pax'],
                                                  test_size=0.2,
                                                  random_state=1024)
```

In [332]:

```
# Merge train sets
train = pd.merge(y_train, X_train, left_index=True, right_index=True)
train.shape
```

Out[332]:

```
(147, 22)
```

In [333]:

```
# Merge test sets
test = pd.merge(y_test, X_test, left_index=True, right_index=True)
test.shape
```

Out[333]:

```
(37, 22)
```

1. Target 'rev_pax'

In [334]:

```
lm1 = smf.ols(formula = "ln_rev_pax ~ ln_Avg_gdp_ori + ln_Avg_gdp_des + ln_Avg_pop_c
              + ln_distance + ln_Avg_Fare + C(Domestic)+ C(less_than_500)", data = t
result1 = lm1.fit()
print (result1.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:              ln_rev_pax   R-squared:
0.978
Model:                             OLS   Adj. R-squared:
0.977
Method:                  Least Squares   F-statistic:
776.5
Date:                 Tue, 04 Oct 2022   Prob (F-statistic):
1.01e-110
Time:                         16:08:45   Log-Likelihood:
118.88
No. Observations:                 147   AIC:
-219.8
Df Residuals:                     138   BIC:
-192.8
Df Model:                           8
Covariance Type:             nonrobust
==============================================================================
======================
                               coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------------
Intercept                     1.8785      0.449      4.188      0.000
0.992       2.765
C(Domestic)[T.1.0]           -0.4585      0.061     -7.550      0.000
-0.579      -0.338
C(less_than_500)[T.1.0]      -0.0650      0.034     -1.885      0.062
-0.133       0.003
ln_Avg_gdp_ori               -0.0870      0.025     -3.446      0.001
-0.137      -0.037
ln_Avg_gdp_des                0.0086      0.024      0.364      0.716
-0.038       0.056
ln_Avg_pop_ori                0.0529      0.012      4.546      0.000
0.030       0.076
ln_Avg_pop_des                0.0235      0.012      1.968      0.051
-0.000       0.047
ln_distance                  -0.0026      0.025     -0.103      0.918
-0.052       0.047
ln_Avg_Fare                   0.6028      0.029     20.574      0.000
0.545       0.661
==============================================================================
========
Omnibus:                        7.533   Durbin-Watson:
1.941
Prob(Omnibus):                  0.023   Jarque-Bera (JB):
11.603
Skew:                           0.215   Prob(JB):
0.00302
Kurtosis:                       4.308   Cond. No.
1.26e+03
==============================================================================
```

```
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.26e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

Evaluation

In [326]:

```
def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual)) * 100
```

In [335]:

```
pred = np.exp(result1.predict(test))
actual = y_test
```

In [336]:

```
mape(actual, pred)
```

Out[336]:

```
157.92850654692526
```

2. Target pax

In [338]:

```
X_train,X_test,y_train,y_test  = train_test_split(sub.drop('rev_pax',axis=1),
                                                  sub['rev_pax'],
                                                  test_size=0.2,
                                                  random_state=1024)

train = pd.merge(y_train, X_train, left_index=True, right_index=True)
test = pd.merge(y_test, X_test, left_index=True, right_index=True)
```

In [339]:

```
lm2 = smf.ols(formula = "ln_pax ~ ln_Avg_gdp_ori + ln_Avg_gdp_des + ln_Avg_pop_ori +
             + ln_distance + ln_Avg_Fare + C(Domestic)+ C(less_than_500)", data = t
result2 = lm2.fit()
print (result2.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                  ln_pax   R-squared:
0.687
Model:                            OLS   Adj. R-squared:
0.669
Method:                 Least Squares   F-statistic:
37.85
Date:                Tue, 04 Oct 2022   Prob (F-statistic):
3.10e-31
Time:                        16:09:29   Log-Likelihood:
-188.38
No. Observations:                 147   AIC:
394.8
Df Residuals:                     138   BIC:
421.7
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
======================
                               coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------------
Intercept                    0.8073      3.627      0.223      0.824
-6.364       7.979
C(Domestic)[T.1.0]          -0.7191      0.491     -1.464      0.145
-1.690       0.252
C(less_than_500)[T.1.0]     -0.2660      0.279     -0.954      0.342
-0.817       0.285
ln_Avg_gdp_ori               1.0830      0.204      5.303      0.000
0.679       1.487
ln_Avg_gdp_des              -0.2760      0.192     -1.437      0.153
-0.656       0.104
ln_Avg_pop_ori               0.1685      0.094      1.793      0.075
-0.017       0.354
ln_Avg_pop_des               0.3849      0.097      3.988      0.000
0.194       0.576
ln_distance                 -0.1700      0.204     -0.835      0.405
-0.573       0.233
ln_Avg_Fare                 -1.7831      0.237     -7.526      0.000
-2.252      -1.315
==============================================================================
========
Omnibus:                       22.217   Durbin-Watson:
1.760
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
30.678
Skew:                           0.838   Prob(JB):
2.18e-07
Kurtosis:                       4.483   Cond. No.
1.26e+03
==============================================================================
```

```
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.26e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

Evaluation

In [340]:

```
pred = np.exp(result2.predict(test))
actual = y_test
```

In [341]:

```
mape(actual, pred)
```

Out[341]:

```
1102.0206049742815
```

Model 2: Service-level

In [281]:

```
sub2 = freq_rt[['rev_pax', 'revenue','pax',
                'Avg_Fare','distance','fare_per_dist','Num_Flights',
                'Domestic','less_than_500']]
```

In [283]:

```
sub2["ln_rev_pax"] = np.log(sub2["rev_pax"])
sub2["ln_pax"] = np.log(sub2["pax"])

sub2["ln_distance"] = np.log(sub2["distance"])
sub2["ln_Avg_Fare"] = np.log(sub2["Avg_Fare"])
sub2["ln_Num_Flights"] = np.log(sub2["Num_Flights"])
```

In [ ]:

```

```

In [284]:

```
lm3 = smf.ols(formula = "ln_pax ~ + ln_distance + ln_Avg_Fare + ln_Num_Flights +\
             C(Domestic)+ C(less_than_500)", data = sub2)
result3 = lm3.fit()
print (result3.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                  ln_pax   R-squared:
0.340
Model:                            OLS   Adj. R-squared:
0.339
Method:                 Least Squares   F-statistic:
364.5
Date:                Tue, 04 Oct 2022   Prob (F-statistic):
5.89e-316
Time:                        15:29:30   Log-Likelihood:
-5529.2
No. Observations:                3547   AIC:
1.107e+04
Df Residuals:                    3541   BIC:
1.111e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
======================
                               coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
---------------------
Intercept                   12.4746      0.198     62.862      0.000
12.086      12.864
C(Domestic)[T.1.0]         -0.6764      0.075     -9.033      0.000
-0.823      -0.530
C(less_than_500)[T.1.0]    -0.2432      0.077     -3.157      0.002
-0.394      -0.092
ln_distance                -0.2606      0.041     -6.343      0.000
-0.341      -0.180
ln_Avg_Fare                -1.1371      0.052    -21.934      0.000
-1.239      -1.035
ln_Num_Flights              0.3755      0.030     12.650      0.000
0.317       0.434
==============================================================================
========
Omnibus:                       23.268   Durbin-Watson:
1.535
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
16.861
Skew:                           0.046   Prob(JB):
0.000218
Kurtosis:                       2.675   Cond. No.
108.
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [ ]:

```
```