In [1]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Load arrivals and departures data

In [2]:

```python
arrivals = pd.read_csv("data/arrivals.csv", sep='\t')
arrivals.head()
```

Out[2]:

	mode	flight_number	callsign	aircraft_model_code	aircraft_model_description	aircra
0	arrivals	JL41	JAL41	B788	Boeing 787-8 Dreamliner	
1	arrivals	SA234	SAA234	A333	Airbus A330-343	
2	arrivals	QF1	QFA1	A388	Airbus A380-842	
3	arrivals	BI3	RBA003	B788	Boeing 787-8 Dreamliner	
4	arrivals	VS4	VIR4C	A333	Airbus A330-343	

5 rows × 22 columns

In [3]:

```
arrivals = arrivals.rename(columns={'flight_duaration':'flight_duration'})
arrivals.head()
```

Out[3]:

	mode	flight_number	callsign	aircraft_model_code	aircraft_model_description	aircraft_
0	arrivals	JL41	JAL41	B788	Boeing 787-8 Dreamliner	
1	arrivals	SA234	SAA234	A333	Airbus A330-343	
2	arrivals	QF1	QFA1	A388	Airbus A380-842	
3	arrivals	BI3	RBA003	B788	Boeing 787-8 Dreamliner	
4	arrivals	VS4	VIR4C	A333	Airbus A330-343	

5 rows × 22 columns

In [4]:

```
arrivals.isnull().sum(axis=0) # check for missing values
```

Out[4]:

```
mode                                0
flight_number                       0
callsign                         6460
aircraft_model_code                72
aircraft_model_description       4625
aircraft_registration            4553
airline_name                        0
airline_iata                        0
airline_icao                        0
flight_origin_code_iata             0
flight_origin_code_icao            97
flight_origin_name                  0
flight_origin_time_offset           0
flight_destination_code_iata        0
flight_destination_code_icao        0
flight_destination_name             0
flight_destination_time_offset      0
flight_departure_scheduled          0
flight_departure_real            7083
flight_arrival_scheduled            0
flight_arrival_real             10327
flight_duration                 10785
dtype: int64
```

In [5]:

```
departures = pd.read_csv("data/departures.csv", sep='\t')
departures.head()
```

Out[5]:

	mode	flight_number	callsign	aircraft_model_code	aircraft_model_description	aircraft_
0	arrivals	JL41	JAL41	B788	Boeing 787-8 Dreamliner	
1	arrivals	SA234	SAA234	A333	Airbus A330-343	
2	arrivals	QF1	QFA1	A388	Airbus A380-842	
3	arrivals	BI3	RBA003	B788	Boeing 787-8 Dreamliner	
4	arrivals	VS4	VIR4C	A333	Airbus A330-343	

5 rows × 22 columns

In [6]:

```
departures = departures.rename(columns={'flight_duaration':'flight_duration'})
departures.head()
```

Out[6]:

	mode	flight_number	callsign	aircraft_model_code	aircraft_model_description	aircra
0	arrivals	JL41	JAL41	B788	Boeing 787-8 Dreamliner	
1	arrivals	SA234	SAA234	A333	Airbus A330-343	
2	arrivals	QF1	QFA1	A388	Airbus A380-842	
3	arrivals	BI3	RBA003	B788	Boeing 787-8 Dreamliner	
4	arrivals	VS4	VIR4C	A333	Airbus A330-343	

5 rows × 22 columns

In [7]:

```
departures.isnull().sum(axis=0) # check for missing values
```

Out[7]:

```
mode                                  0
flight_number                         0
callsign                          12316
aircraft_model_code                 145
aircraft_model_description         9103
aircraft_registration              8958
airline_name                          0
airline_iata                          0
airline_icao                          0
flight_origin_code_iata               0
flight_origin_code_icao              97
flight_origin_name                    0
flight_origin_time_offset             0
flight_destination_code_iata          0
flight_destination_code_icao        111
flight_destination_name               0
flight_destination_time_offset        0
flight_departure_scheduled            0
flight_departure_real             13262
flight_arrival_scheduled              0
flight_arrival_real               22013
flight_duration                  147423
dtype: int64
```

Load weather data

In [8]:

```
import datetime

weather = pd.read_csv("data/METAR_all_airports.csv")
weather['timestamp'] = weather['UTC DATE/TIME'].map(lambda t: datetime.datetim
e.strptime(t,'%Y-%m-%d %H:%M:%S').timestamp())
weather['hours'] = weather['timestamp'].map(lambda s: np.rint(s/3600)).astype(
int)
weather.head()
```

Out[8]:

	AIRPORT_IATA	AIRPORT_ICAO	UTC DATE/TIME	METAR	DESCRIPTION	timestamp	h
0	LHR	EGLL	2019-05-18 23:50:00	EGLL 182350Z AUTO 01003KT 9999 NCD 11/09 Q1009...	Day: 18th Time: 23:50 UTC Wind direction: 10 W...	1.558220e+09	43
1	LHR	EGLL	2019-05-18 23:20:00	EGLL 182320Z AUTO 02002KT 9999 NCD 10/08 Q1009...	Day: 18th Time: 23:20 UTC Wind direction: 20 W...	1.558218e+09	43
2	LHR	EGLL	2019-05-18 22:50:00	EGLL 182250Z AUTO 04002KT 9999 NCD 12/09 Q1009...	Day: 18th Time: 22:50 UTC Wind direction: 40 W...	1.558216e+09	43
3	LHR	EGLL	2019-05-18 22:20:00	EGLL 182220Z AUTO 06002KT 9999 NCD 12/09 Q1008	Day: 18th Time: 22:20 UTC Wind direction: 60 W...	1.558214e+09	43
4	LHR	EGLL	2019-05-18 21:50:00	EGLL 182150Z AUTO VRB02KT 9999 NCD 13/09 Q1008...	Day: 18th Time: 21:50 UTC Wind speed: 2kt Temp...	1.558213e+09	43

In [9]:

```
weather.isnull().sum(axis=0) # check for missing values
```

Out[9]:

```
AIRPORT_IATA      0
AIRPORT_ICAO      0
UTC DATE/TIME     0
METAR             0
DESCRIPTION       0
timestamp         0
hours             0
dtype: int64
```

In [10]:

```
airport_weather = weather.groupby(['AIRPORT_IATA','hours']).agg({'METAR':'last
','DESCRIPTION':'last'}).reset_index()
airport_weather.head()
```

Out[10]:

	AIRPORT_IATA	hours	METAR	DESCRIPTION
0	ATH	432815	LGAV 180020Z VRB01KT CAVOK 13/10 Q1009 NOSIG	Day: 18th Time: 00:20 UTC Wind speed: 1kt Temp...
1	ATH	432816	LGAV 180050Z 20002KT CAVOK 13/10 Q1009 NOSIG	Day: 18th Time: 00:50 UTC Wind direction: 200 ...
2	ATH	432817	LGAV 180150Z VRB01KT CAVOK 13/11 Q1009 NOSIG	Day: 18th Time: 01:50 UTC Wind speed: 1kt Temp...
3	ATH	432818	LGAV 180250Z VRB01KT CAVOK 13/11 Q1010 NOSIG	Day: 18th Time: 02:50 UTC Wind speed: 1kt Temp...
4	ATH	432819	LGAV 180350Z 00000KT CAVOK 12/10 Q1009 NOSIG	Day: 18th Time: 03:50 UTC Wind direction: 0 Wi...

In [11]:

```
airport_weather.isnull().sum(axis=0) # check for missing values
```

Out[11]:

```
AIRPORT_IATA     0
hours            0
METAR            0
DESCRIPTION      0
dtype: int64
```

Calculate flight delays and cancellations

In [12]:

```
flights = pd.concat ([arrivals,departures], sort=False).drop_duplicates()
flights.isnull().sum(axis=0) # re-check for missing values
```

Out[12]:

```
mode                                 0
flight_number                        0
callsign                         12316
aircraft_model_code                145
aircraft_model_description        9103
aircraft_registration             8958
airline_name                         0
airline_iata                         0
airline_icao                         0
flight_origin_code_iata              0
flight_origin_code_icao             97
flight_origin_name                   0
flight_origin_time_offset            0
flight_destination_code_iata         0
flight_destination_code_icao       111
flight_destination_name              0
flight_destination_time_offset       0
flight_departure_scheduled           0
flight_departure_real            13262
flight_arrival_scheduled             0
flight_arrival_real              22013
flight_duration                 147423
dtype: int64
```

In [13]:

```
flights['assumed_cancellation'] = np.where(flights['flight_departure_real'].is
null(),1,0).astype(int)
flights['assumed_cancellation'].describe()
```

Out[13]:

```
count    282133.000000
mean          0.047006
std           0.211652
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           1.000000
Name: assumed_cancellation, dtype: float64
```

In [14]:

```
flights['flight_arrival_real'] = np.where(flights['flight_arrival_real'].isnul
l(),
                                          flights['flight_arrival_scheduled'],
                                          flights['flight_arrival_real'])
flights['scheduled_arrival_hour'] = flights['flight_arrival_scheduled'].map(la
mbda s: np.rint(s/3600)).astype(int)
```

In [15]:

```
flights['flight_departure_real'] = np.where(flights['flight_departure_real'].i
snull(),
                                            flights['flight_departure_scheduled'
],
                                            flights['flight_departure_real'])
flights['scheduled_departure_hour'] = flights['flight_departure_scheduled'].ma
p(lambda s: np.rint(s/3600)).astype(int)
```

In [16]:

```
flights['flight_duration_scheduled'] = flights['flight_arrival_scheduled'] - f
lights['flight_departure_scheduled']

flights['flight_duration'] = np.where(flights['flight_duration'].isnull(),
                                      flights['flight_duration_scheduled']
,
                                      flights['flight_duration'])
```

In [17]:

```
flights['callsign'].fillna('Missing', inplace=True)
flights['aircraft_model_code'].fillna('Missing', inplace=True)
flights['aircraft_model_description'].fillna('Missing', inplace=True)
flights['aircraft_registration'].fillna('Missing', inplace=True)
flights['flight_origin_code_icao'].fillna('Missing', inplace=True)
flights['flight_destination_code_icao'].fillna('Missing', inplace=True)
```

In [18]:

```
flights.isnull().sum(axis=0) # re-check for missing values
```

Out[18]:

```
mode                              0
flight_number                     0
callsign                          0
aircraft_model_code               0
aircraft_model_description        0
aircraft_registration             0
airline_name                      0
airline_iata                      0
airline_icao                      0
flight_origin_code_iata           0
flight_origin_code_icao           0
flight_origin_name                0
flight_origin_time_offset         0
flight_destination_code_iata      0
flight_destination_code_icao      0
flight_destination_name           0
flight_destination_time_offset    0
flight_departure_scheduled        0
flight_departure_real             0
flight_arrival_scheduled          0
flight_arrival_real               0
flight_duration                   0
assumed_cancellation              0
scheduled_arrival_hour            0
scheduled_departure_hour          0
flight_duration_scheduled         0
dtype: int64
```

In [19]:

```
flights['arrival_delay'] = np.where(flights['flight_arrival_real'] < flights['flight_arrival_scheduled'],0,
                                    flights['flight_arrival_real'] - flights['flight_arrival_scheduled'])
flights['arrival_delay'].describe()
```

Out[19]:

```
count    282133.000000
mean        707.976486
std        2289.777189
min           0.000000
25%           0.000000
50%           0.000000
75%         386.000000
max       85290.000000
Name: arrival_delay, dtype: float64
```

In [20]:

```
flights['departure_delay'] = np.where(flights['flight_departure_real'] < fligh
ts['flight_departure_scheduled'],0,
                                      flights['flight_departure_real'] - fligh
ts['flight_departure_scheduled'])
flights['departure_delay'].describe()
```

Out[20]:

```
count    282133.000000
mean       1818.733222
std        2853.318770
min           0.000000
25%         612.000000
50%        1134.000000
75%        2024.000000
max      139440.000000
Name: departure_delay, dtype: float64
```

Derive features from the categorical data

In [21]:

```
origin_delays = flights.groupby('flight_origin_name').agg({'departure_delay':'
mean',
                                                             'assumed_cancellati
on':'mean'}).rename(
    columns={'departure_delay':'origin_departure_delay',
                       'assumed_cancellation':'origin_cancellation'})

origin_delays.dtypes
```

Out[21]:

```
origin_departure_delay    float64
origin_cancellation       float64
dtype: object
```

In [22]:

```
origin_delays.sort_values(by='origin_departure_delay', ascending=False).head()
```

Out[22]:

flight_origin_name	origin_departure_delay	origin_cancellation
Angeles City Clark International Airport	21897.000000	0.000000
Rutland Southern Vermont Regional Airport	19046.000000	0.000000
Cold Lake	14233.000000	0.000000
Salisbury Ocean City Wicomico Regional Airport	11314.000000	0.000000
Maastricht Aachen Airport	9493.428571	0.714286

In [23]:

```
destination_delays = flights.groupby('flight_destination_name').agg({'arrival_
delay':'mean'}).rename(
    columns={'arrival_delay':'destination_arrival_delay'})

destination_delays.dtypes
```

Out[23]:

```
destination_arrival_delay    float64
dtype: object
```

In [24]:

```
destination_delays.sort_values(by='destination_arrival_delay', ascending=False
).head()
```

Out[24]:

flight_destination_name	destination_arrival_delay
Lake City Gateway Airport	14857.000
Millington Regional Jetport	13437.000
Vitoria Eurico de Aguiar Salles Airport	9273.000
Kassel Calden Airport	8569.625
Jacksonville Cecil Airport	8514.000

In [25]:

```
airline_delays = flights.groupby('airline_name').agg({'departure_delay':'mean'
,'arrival_delay':'mean',
                                                       'assumed_cancellation':'m
ean'}).rename(
columns={'departure_delay':'airline_departure_delay','arrival_delay':'airline_
arrival_delay',
        'assumed_cancellation':'airline_cancellation'})

airline_delays.dtypes
```

Out[25]:

```
airline_departure_delay    float64
airline_arrival_delay      float64
airline_cancellation       float64
dtype: object
```

In [26]:

```
airline_delays.sort_values(by='airline_departure_delay', ascending=False).head
()
```

Out[26]:

airline_name	airline_departure_delay	airline_arrival_delay	airline_cancellation
El Al Israel Cargo	25850.00	24160.0	0.000
Magma Aviation	18494.00	312.5	0.000
Frontier (Seymour the Walrus Livery)	14423.50	12360.0	0.000
Nolinor Aviation	14066.25	0.0	0.375
Norwegian (Thor Heyerdahl livery)	13890.00	0.0	0.000

In [27]:

```
airline_delays.sort_values(by='airline_arrival_delay', ascending=False).head()
```

Out[27]:

airline_name	airline_departure_delay	airline_arrival_delay	airline_cancellation
El Al Israel Cargo	25850.00000	24160.000000	0.000000
Frontier (Seymour the Walrus Livery)	14423.50000	12360.000000	0.000000
Cargolux (Cutaway Livery)	11677.00000	9461.500000	0.250000
Aviolet	9727.00000	8299.000000	0.000000
Ethiopian Cargo	6943.62963	8209.777778	0.074074

In [28]:

```
aircraft_delays = flights.groupby('aircraft_model_code').agg({'departure_delay
':'mean','arrival_delay':'mean',
                                                                     'assumed_c
ancellation':'mean'}).rename(
columns={'departure_delay':'aircraft_departure_delay','arrival_delay':'aircraf
t_arrival_delay',
        'assumed_cancellation':'aircraft_cancellation'})

aircraft_delays.dtypes
```

Out[28]:

```
aircraft_departure_delay    float64
aircraft_arrival_delay      float64
aircraft_cancellation       float64
dtype: object
```

In [29]:

```
aircraft_delays.sort_values(by='aircraft_arrival_delay', ascending=False).head
()
```

Out[29]:

aircraft_model_code	aircraft_departure_delay	aircraft_arrival_delay	aircraft_cancellation
SW4	2280.666667	8937.000000	0.00000
C152	4898.000000	5899.000000	0.00000
0000	7057.833333	5795.833333	0.00000
LJ35	5589.428571	3383.142857	0.00000
A30B	3503.281250	3338.062500	0.15625

In [30]:

```
flights = pd.merge (flights, origin_delays, on='flight_origin_name', suffixes=
('_flight','_origin'))
flights = pd.merge (flights, destination_delays, on='flight_destination_name',
suffixes=('_flight','_destination'))
flights = pd.merge (flights, airline_delays, on='airline_name', suffixes=('_fl
ight','_airline'))
flights = pd.merge (flights, aircraft_delays, on='aircraft_model_code', suffix
es=('_flight','_aircraft'))
flights.dtypes
```

Out[30]:

```
mode                              object
flight_number                     object
callsign                          object
aircraft_model_code               object
aircraft_model_description        object
aircraft_registration             object
airline_name                      object
airline_iata                      object
airline_icao                      object
flight_origin_code_iata           object
flight_origin_code_icao           object
flight_origin_name                object
flight_origin_time_offset          int64
flight_destination_code_iata      object
flight_destination_code_icao      object
flight_destination_name           object
flight_destination_time_offset     int64
flight_departure_scheduled         int64
flight_departure_real            float64
flight_arrival_scheduled           int64
flight_arrival_real              float64
flight_duration                  float64
assumed_cancellation               int64
scheduled_arrival_hour             int64
scheduled_departure_hour           int64
flight_duration_scheduled          int64
arrival_delay                    float64
departure_delay                  float64
origin_departure_delay           float64
origin_cancellation              float64
destination_arrival_delay        float64
airline_departure_delay          float64
airline_arrival_delay            float64
airline_cancellation             float64
aircraft_departure_delay         float64
aircraft_arrival_delay           float64
aircraft_cancellation            float64
dtype: object
```

In [31]:

```
flights.isnull().sum(axis=0) # re-check for missing values
```

Out[31]:

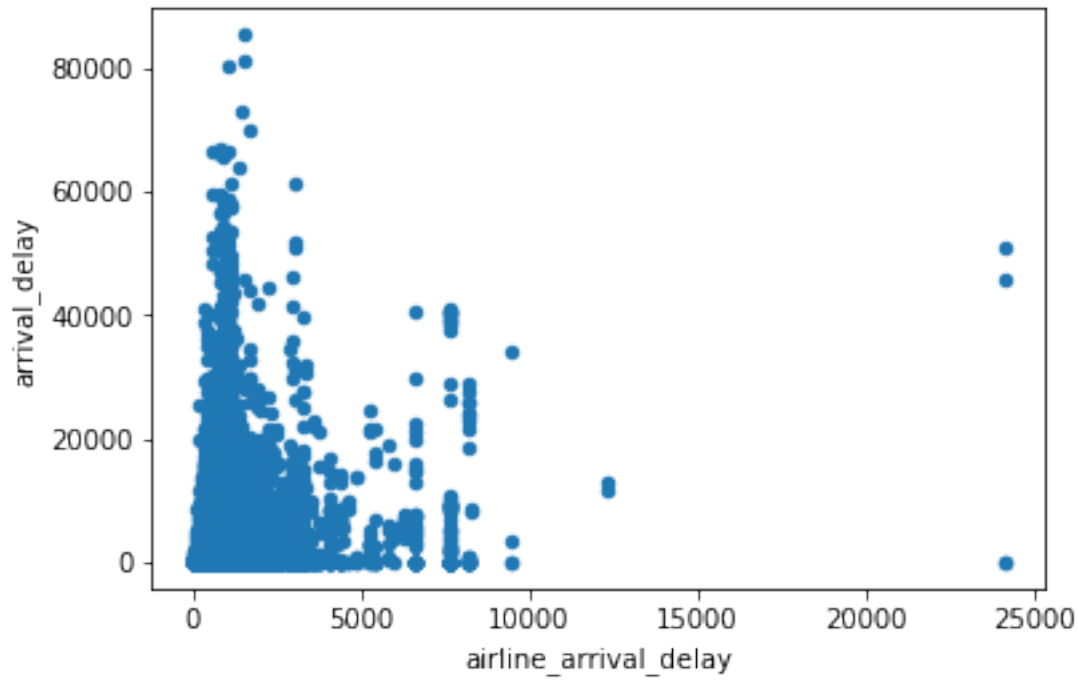
```
mode                              0
flight_number                     0
callsign                          0
aircraft_model_code               0
aircraft_model_description        0
aircraft_registration             0
airline_name                      0
airline_iata                      0
airline_icao                      0
flight_origin_code_iata           0
flight_origin_code_icao           0
flight_origin_name                0
flight_origin_time_offset         0
flight_destination_code_iata      0
flight_destination_code_icao      0
flight_destination_name           0
flight_destination_time_offset    0
flight_departure_scheduled        0
flight_departure_real             0
flight_arrival_scheduled          0
flight_arrival_real               0
flight_duration                   0
assumed_cancellation              0
scheduled_arrival_hour            0
scheduled_departure_hour          0
flight_duration_scheduled         0
arrival_delay                     0
departure_delay                   0
origin_departure_delay            0
origin_cancellation               0
destination_arrival_delay         0
airline_departure_delay           0
airline_arrival_delay             0
airline_cancellation              0
aircraft_departure_delay          0
aircraft_arrival_delay            0
aircraft_cancellation             0
dtype: int64
```

In [32]:

```
flights.plot ('airline_arrival_delay', 'arrival_delay', kind='scatter')
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1057ef160>
```



In [33]:

```
from sklearn.metrics import explained_variance_score, r2_score

print (explained_variance_score (flights['arrival_delay'], flights['airline_ar
rival_delay']))
```

```
0.0289657022354306
```

In [34]:

```
print (r2_score (flights['arrival_delay'], flights['airline_arrival_delay']))
```

```
0.028965702235430713
```

In [35]:

```
print (explained_variance_score (flights['arrival_delay'], flights['aircraft_a
rrival_delay']))
```

```
0.011183536476836009
```

In [36]:

```
print (r2_score (flights['arrival_delay'], flights['aircraft_arrival_delay']))
```

```
0.011183536476835898
```

In [37]:

```
print (explained_variance_score (flights['arrival_delay'], flights['destination_arrival_delay']))
```

0.01943271733567653

In [38]:

```
print (explained_variance_score (flights['departure_delay'], flights['origin_departure_delay']))
```

0.03507124920176963

In [39]:

```
print (explained_variance_score (flights['assumed_cancellation'], flights['origin_cancellation']))
```

0.07887898985493258

In [42]:

```
print (explained_variance_score (flights['assumed_cancellation'], flights['airline_cancellation']))
```

0.21007337063634002

In [44]:

```
print (explained_variance_score (flights['assumed_cancellation'], flights['aircraft_cancellation']))
```

0.5538384901480515

In [46]:

```
print (r2_score (flights['assumed_cancellation'], flights['aircraft_cancellation']))
```

0.5538384901480515

Derive features from the weather data

In [47]:

```
flight_weather = pd.merge(flights,airport_weather,
                          left_on=['flight_origin_code_iata','scheduled_depart
ure_hour'],
                          right_on=['AIRPORT_IATA','hours'],
                          suffixes=('_origin','_departure')).rename(columns={'
METAR':'departure_METAR',
                                                                             '
DESCRIPTION':'departure_weather'})
flight_weather = pd.merge(flight_weather,airport_weather,
                          left_on=['flight_destination_code_iata','scheduled_a
rrival_hour'],
                          right_on=['AIRPORT_IATA','hours'],
                          suffixes=('_destination','_arrival')).rename(columns
={'METAR':'arrival_METAR',
                                                                             '
DESCRIPTION':'arrival_weather'})
flight_weather.head()
```

Out[47]:

	mode	flight_number	callsign	aircraft_model_code	aircraft_model_description	airc
0	arrivals	UA900	UAL900	B77W	Boeing 777-322(ER)	
1	departures	UA900	UAL900	B77W	Boeing 777-322(ER)	
2	arrivals	VS19	VIR19Z	B789	Boeing 787-9 Dreamliner	
3	departures	VS19	VIR19Z	B789	Boeing 787-9 Dreamliner	
4	arrivals	BA285	BAW11M	B744	Boeing 747-436	

5 rows × 45 columns

In [48]:

```
flight_weather.isnull().sum(axis=0) # check for missing values
```

Out[48]:

```
mode                              0
flight_number                     0
callsign                          0
aircraft_model_code               0
aircraft_model_description        0
aircraft_registration             0
airline_name                      0
airline_iata                      0
airline_icao                      0
flight_origin_code_iata           0
flight_origin_code_icao           0
flight_origin_name                0
flight_origin_time_offset         0
flight_destination_code_iata      0
flight_destination_code_icao      0
flight_destination_name           0
flight_destination_time_offset    0
flight_departure_scheduled        0
flight_departure_real             0
flight_arrival_scheduled          0
flight_arrival_real               0
flight_duration                   0
assumed_cancellation              0
scheduled_arrival_hour            0
scheduled_departure_hour          0
flight_duration_scheduled         0
arrival_delay                     0
departure_delay                   0
origin_departure_delay            0
origin_cancellation               0
destination_arrival_delay         0
airline_departure_delay           0
airline_arrival_delay             0
airline_cancellation              0
aircraft_departure_delay          0
aircraft_arrival_delay            0
aircraft_cancellation             0
AIRPORT_IATA_destination          0
hours_destination                 0
departure_METAR                   0
departure_weather                 0
AIRPORT_IATA_arrival              0
hours_arrival                     0
arrival_METAR                     0
arrival_weather                   0
dtype: int64
```

In [49]:

```
flight_weather.dtypes
```

Out[49]:

```
mode                               object
flight_number                      object
callsign                           object
aircraft_model_code                object
aircraft_model_description         object
aircraft_registration              object
airline_name                       object
airline_iata                       object
airline_icao                       object
flight_origin_code_iata            object
flight_origin_code_icao            object
flight_origin_name                 object
flight_origin_time_offset           int64
flight_destination_code_iata       object
flight_destination_code_icao       object
flight_destination_name            object
flight_destination_time_offset      int64
flight_departure_scheduled          int64
flight_departure_real             float64
flight_arrival_scheduled            int64
flight_arrival_real               float64
flight_duration                   float64
assumed_cancellation                int64
scheduled_arrival_hour              int64
scheduled_departure_hour            int64
flight_duration_scheduled           int64
arrival_delay                     float64
departure_delay                   float64
origin_departure_delay            float64
origin_cancellation               float64
destination_arrival_delay         float64
airline_departure_delay           float64
airline_arrival_delay             float64
airline_cancellation              float64
aircraft_departure_delay          float64
aircraft_arrival_delay            float64
aircraft_cancellation             float64
AIRPORT_IATA_destination           object
hours_destination                   int64
departure_METAR                    object
departure_weather                  object
AIRPORT_IATA_arrival               object
hours_arrival                       int64
arrival_METAR                      object
arrival_weather                    object
dtype: object
```

In [50]:

```
import re

def extract_visibility (w):
    match = re.search('Visibility: (.+?) m',w)
    if match:
        return int (match.group(1))
    else:
        return 9999 # METAR NOSIG default is 9999 m

flight_weather['departure_visibility'] = flight_weather['departure_weather'].m
ap(lambda w: extract_visibility(w))
flight_weather['arrival_visibility'] = flight_weather['arrival_weather'].map(l
ambda w: extract_visibility(w))

flight_weather['departure_visibility'].describe()
```

Out[50]:

```
count    25539.000000
mean      9912.734719
std        603.695498
min        700.000000
25%       9999.000000
50%       9999.000000
75%       9999.000000
max       9999.000000
Name: departure_visibility, dtype: float64
```

In [51]:

```
print (explained_variance_score (flight_weather['arrival_delay'],flight_weathe
r['arrival_visibility']))
```

```
-0.05240637972192541
```

In [52]:

```
print (explained_variance_score (flight_weather['departure_delay'],flight_weat
her['departure_visibility']))
```

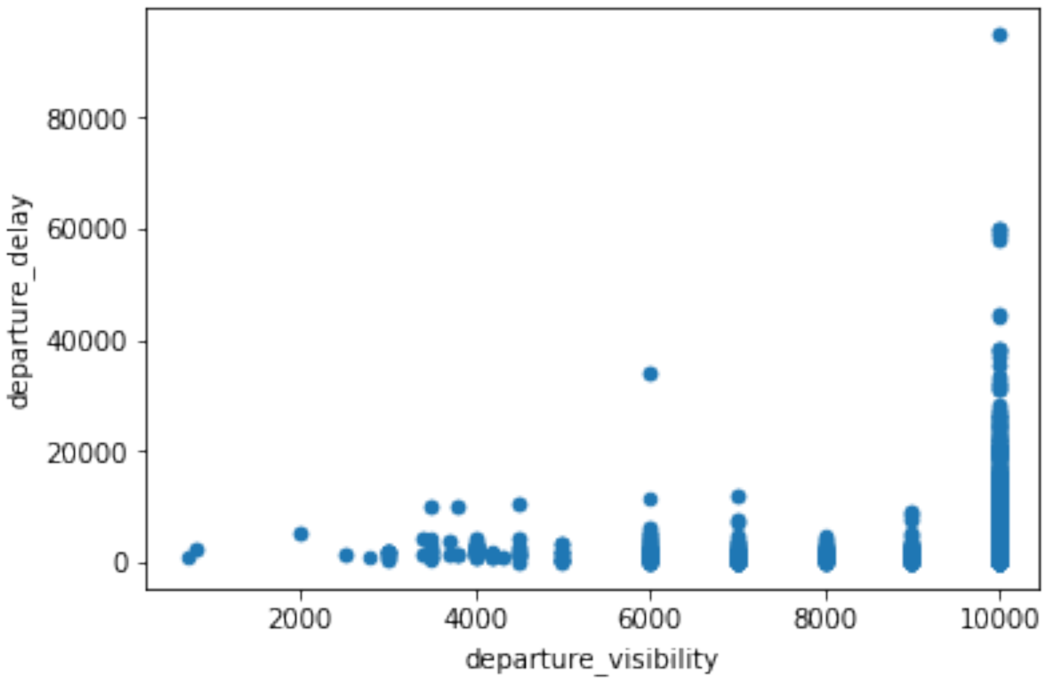
```
-0.04114611289283254
```

In [53]:

```
flight_weather.plot('departure_visibility', 'departure_delay', kind='scatter')
```

Out[53]:

<matplotlib.axes._subplots.AxesSubplot at 0x11e229eb8>

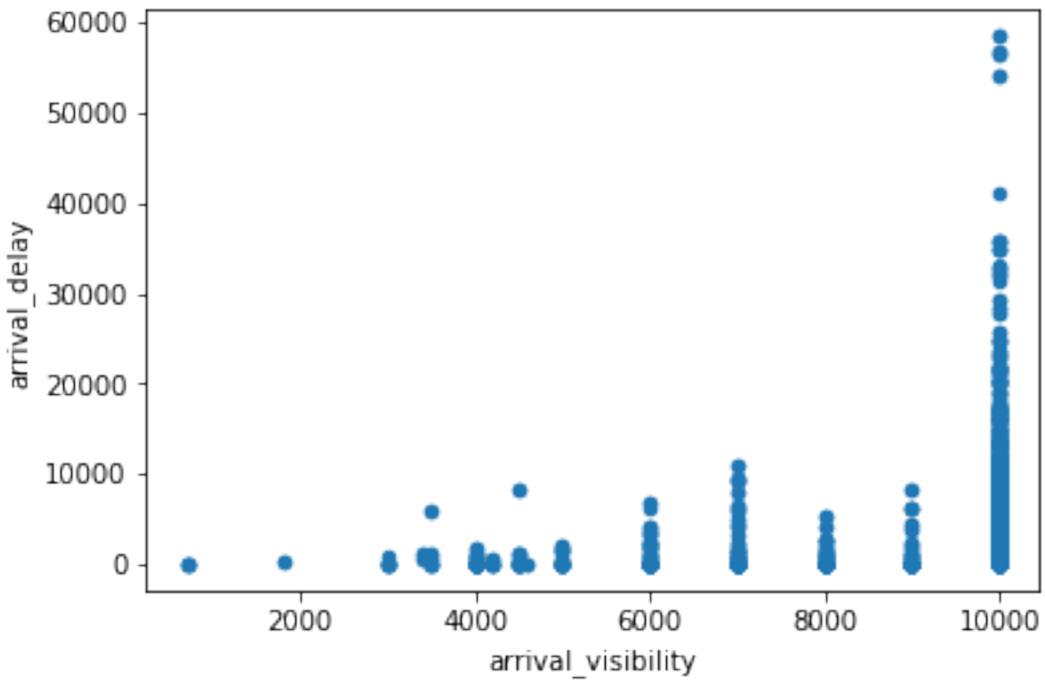


In [54]:

```
flight_weather.plot('arrival_visibility', 'arrival_delay', kind='scatter')
```

Out[54]:

<matplotlib.axes._subplots.AxesSubplot at 0x11e079b00>



In [55]:

```
def extract_temperature (w):
    match = re.search('Temperature: (.+?)C',w)
    if match:
        return int (match.group(1))
    else:
        return 16 # if temperature missing then assume gloval average temperat
ure as default

flight_weather['departure_temperature'] = flight_weather['departure_weather'].
map(lambda w: extract_temperature(w))
flight_weather['arrival_temperature'] = flight_weather['arrival_weather'].map(
lambda w: extract_temperature(w))

flight_weather['departure_temperature'].describe()
```

Out[55]:

```
count    25539.000000
mean        19.097733
std          4.654455
min          6.000000
25%         16.000000
50%         18.000000
75%         22.000000
max         34.000000
Name: departure_temperature, dtype: float64
```

In [56]:

```
def is_freezing (t):
    if (t < 4):
        return 1
    else:
        return 0

flight_weather['freezing_at_departure'] = flight_weather['departure_temperatur
e'].map(lambda t: is_freezing(t))
flight_weather['freezing_at_arrival'] = flight_weather['arrival_temperature'].
map(lambda t: is_freezing(t))

flight_weather['freezing_at_departure'].describe()
```

Out[56]:

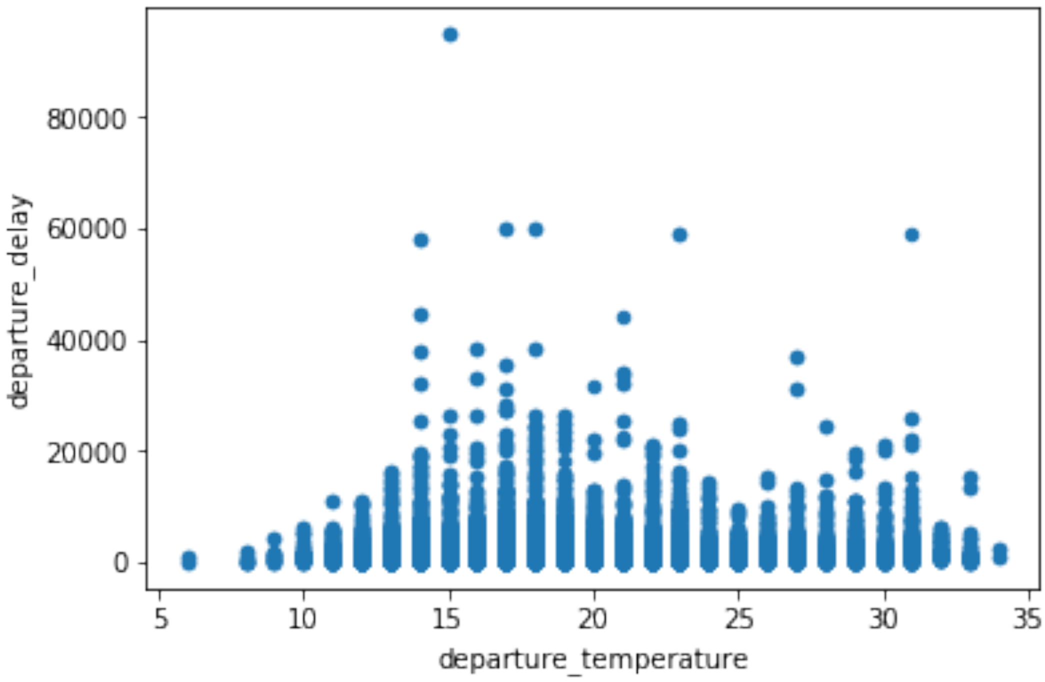
```
count    25539.0
mean         0.0
std          0.0
min          0.0
25%          0.0
50%          0.0
75%          0.0
max          0.0
Name: freezing_at_departure, dtype: float64
```

In [57]:

```
flight_weather.plot('departure_temperature', 'departure_delay', kind='scatter'
)
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e143b38>
```

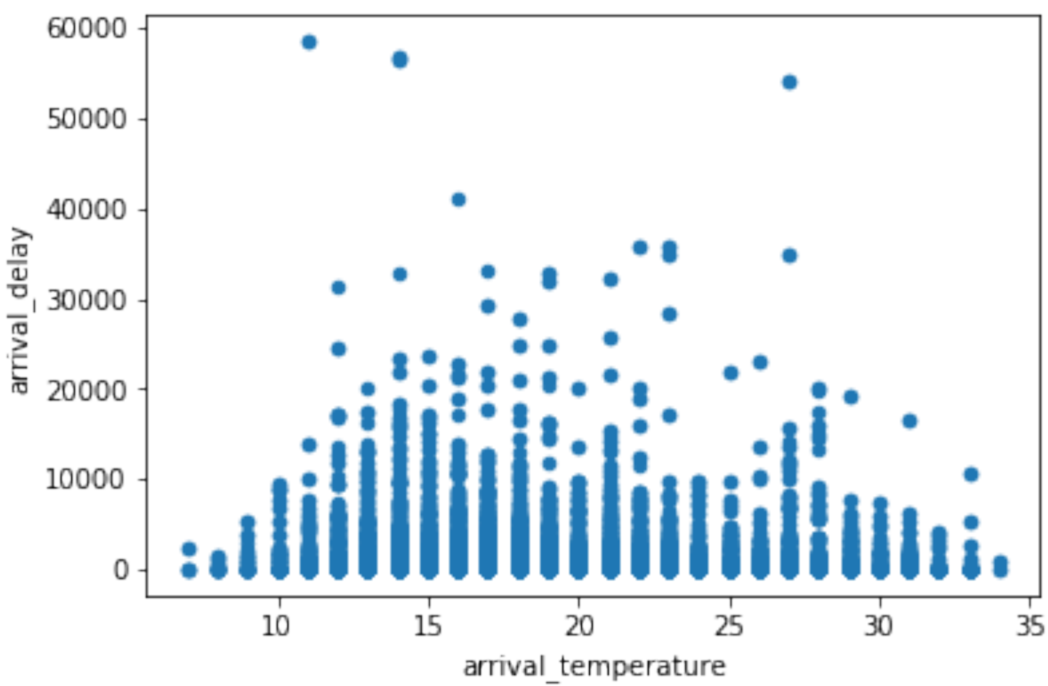


In [58]:

```
flight_weather.plot('arrival_temperature', 'arrival_delay', kind='scatter')
```

Out[58]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e476128>
```



In [59]:

```
def extract_windspeed (w):
    match = re.search('Wind speed: (.+?)kt',w)
    if match:
        return int (match.group(1))
    else:
        return 0 # if windspeed is missing then assume no wind

flight_weather['departure_windspeed'] = flight_weather['departure_weather'].ma
p(lambda w: extract_windspeed(w))
flight_weather['arrival_windspeed'] = flight_weather['arrival_weather'].map(la
mbda w: extract_windspeed(w))

flight_weather['departure_windspeed'].describe()
```

Out[59]:

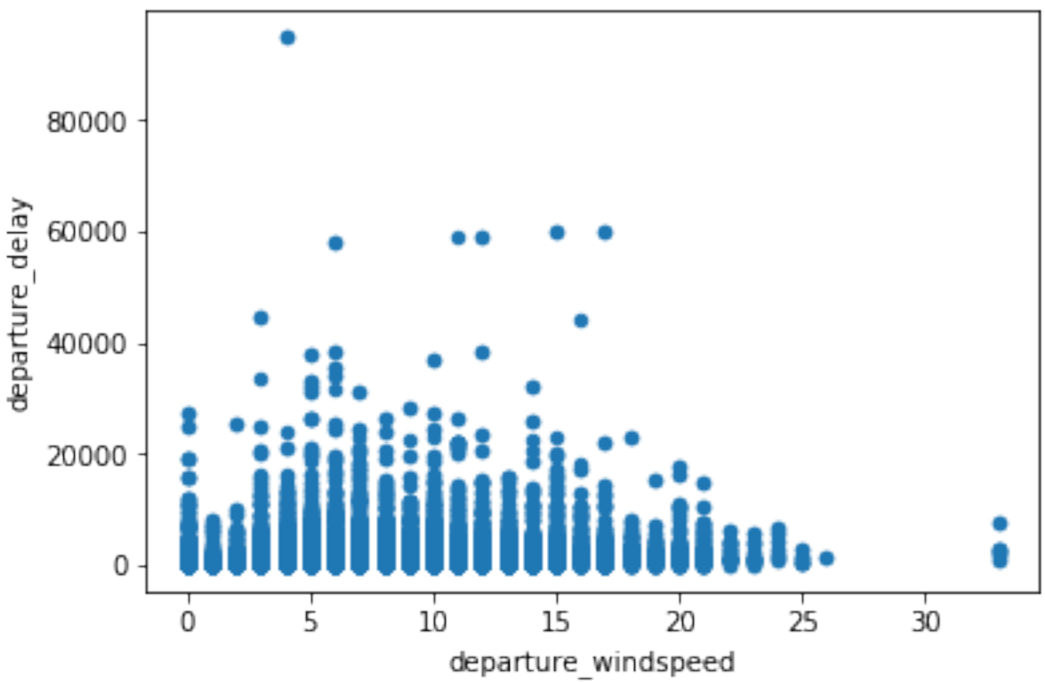
```
count    25539.000000
mean         8.702847
std          4.467975
min          0.000000
25%          6.000000
50%          8.000000
75%         11.000000
max         33.000000
Name: departure_windspeed, dtype: float64
```

In [60]:

```
flight_weather.plot('departure_windspeed', 'departure_delay', kind='scatter')
```

Out[60]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e639048>
```

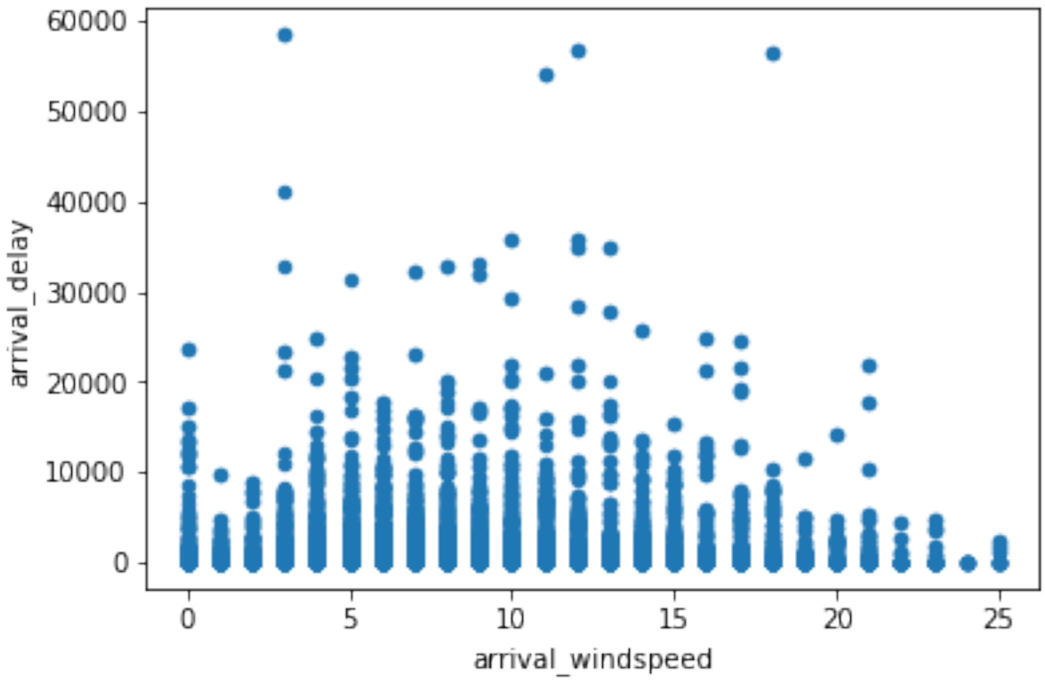


In [61]:

```
flight_weather.plot('arrival_windspeed', 'arrival_delay', kind='scatter')
```

Out[61]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e6f94e0>
```



In [62]:

```
def extract_pressure (w):
    match = re.search('Pressure: (.+?) hPa',w)
    if match:
        return int (match.group(1))
    else:
        return 1000 # if pressure is missing then assume global average

flight_weather['departure_pressure'] = flight_weather['departure_weather'].map
(lambda w: extract_pressure(w))
flight_weather['arrival_pressure'] = flight_weather['arrival_weather'].map(lam
bda w: extract_pressure(w))

flight_weather['departure_pressure'].describe()
```

Out[62]:

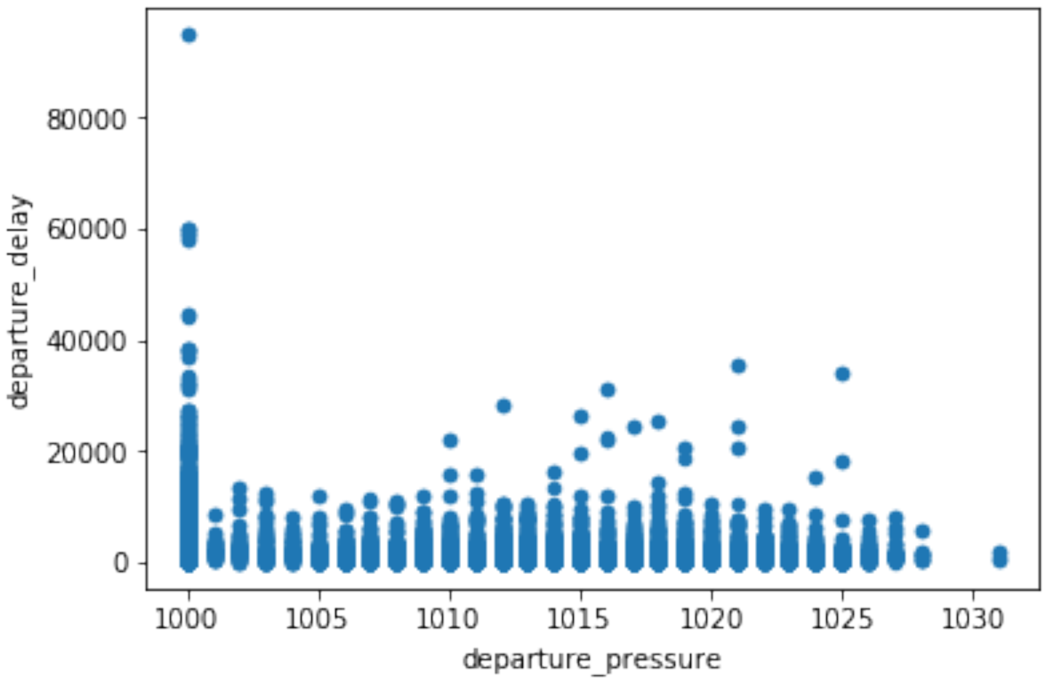
```
count    25539.000000
mean      1007.669838
std          8.642688
min       1000.000000
25%       1000.000000
50%       1001.000000
75%       1015.000000
max       1031.000000
Name: departure_pressure, dtype: float64
```

In [63]:

```
flight_weather.plot('departure_pressure', 'departure_delay', kind='scatter')
```

Out[63]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e7470b8>
```

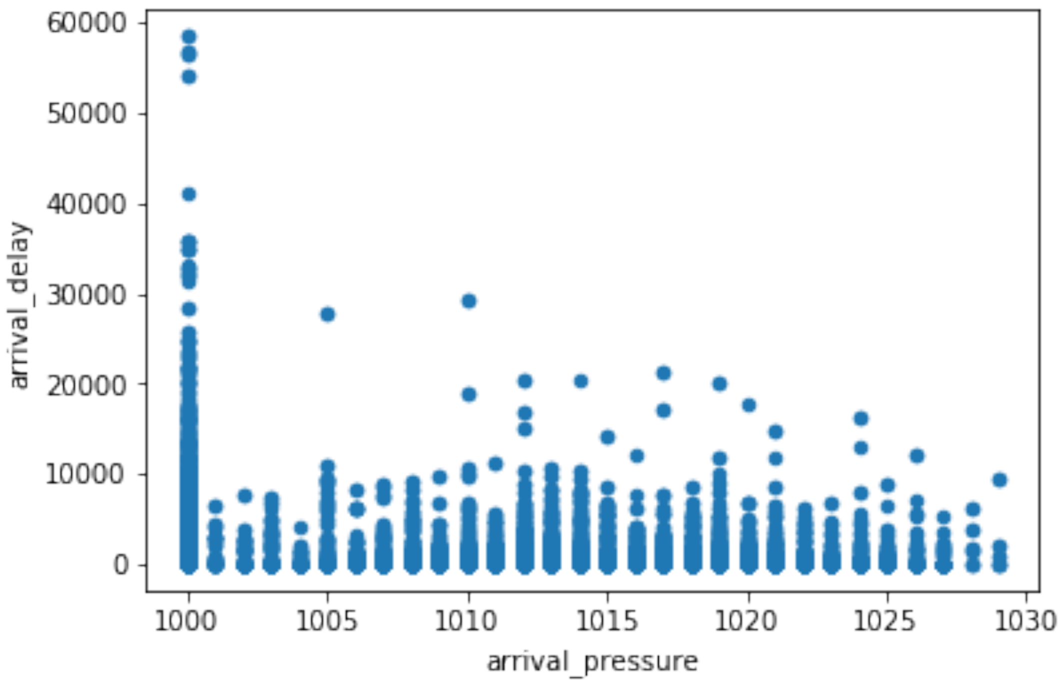


In [64]:

```
flight_weather.plot('arrival_pressure', 'arrival_delay', kind='scatter')
```

Out[64]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11f8e87b8>
```



In [65]:

```
def extract_dewpoint (w):
    match = re.search('Dew point: (.+?)C',w)
    if match:
        return int (match.group(1))
    else:
        return 16 # if temperature missing then global average temperature as default

flight_weather['departure_dewpoint'] = flight_weather['departure_weather'].map(lambda w: extract_dewpoint(w))
flight_weather['arrival_dewpoint'] = flight_weather['arrival_weather'].map(lambda w: extract_dewpoint(w))
flight_weather['departure_clouds'] = flight_weather['departure_temperature'] - flight_weather['departure_dewpoint']
flight_weather['arrival_clouds'] = flight_weather['arrival_temperature'] - flight_weather['arrival_dewpoint']

flight_weather['departure_dewpoint'].describe()
```

Out[65]:

```
count    25539.000000
mean        11.480168
std          4.152691
min         -2.000000
25%          9.000000
50%         11.000000
75%         14.000000
max         25.000000
Name: departure_dewpoint, dtype: float64
```

In [66]:

```
flight_weather['arrival_dewpoint'].describe()
```

Out[66]:

```
count    25539.000000
mean        11.563021
std          4.195023
min         -2.000000
25%          9.000000
50%         11.000000
75%         14.000000
max         25.000000
Name: arrival_dewpoint, dtype: float64
```

In [67]:

```
flight_weather['departure_clouds'].describe()
```

Out[67]:

```
count    25539.000000
mean         7.617565
std          4.614260
min          0.000000
25%          4.000000
50%          7.000000
75%         10.000000
max         24.000000
Name: departure_clouds, dtype: float64
```

In [68]:

```
flight_weather['arrival_clouds'].describe()
```

Out[68]:

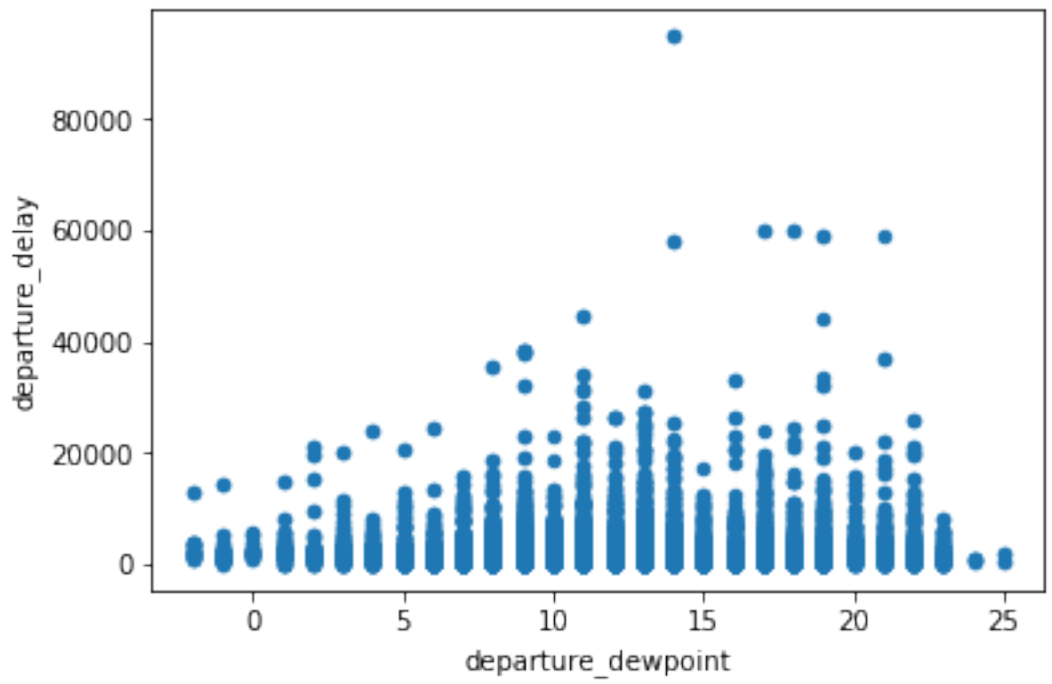
```
count    25539.000000
mean         7.558362
std          4.613021
min          0.000000
25%          4.000000
50%          7.000000
75%         10.000000
max         24.000000
Name: arrival_clouds, dtype: float64
```

In [69]:

```
flight_weather.plot('departure_dewpoint', 'departure_delay', kind='scatter')
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11fe94358>
```

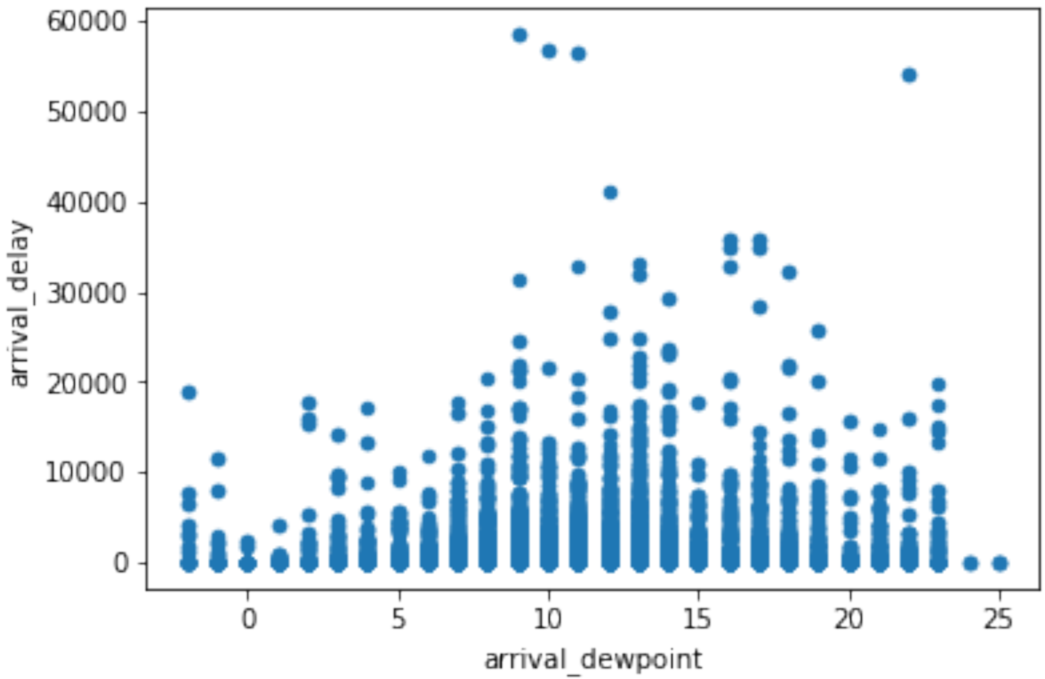


In [70]:

```
flight_weather.plot('arrival_dewpoint', 'arrival_delay', kind='scatter')
```

Out[70]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12001b8d0>
```



In [71]:

```
print (explained_variance_score (flight_weather['departure_delay'],flight_weat
her['departure_dewpoint']))
```

```
0.00018604841301383956
```

In [72]:

```
print (explained_variance_score (flight_weather['arrival_delay'],flight_weathe
r['arrival_dewpoint']))
```

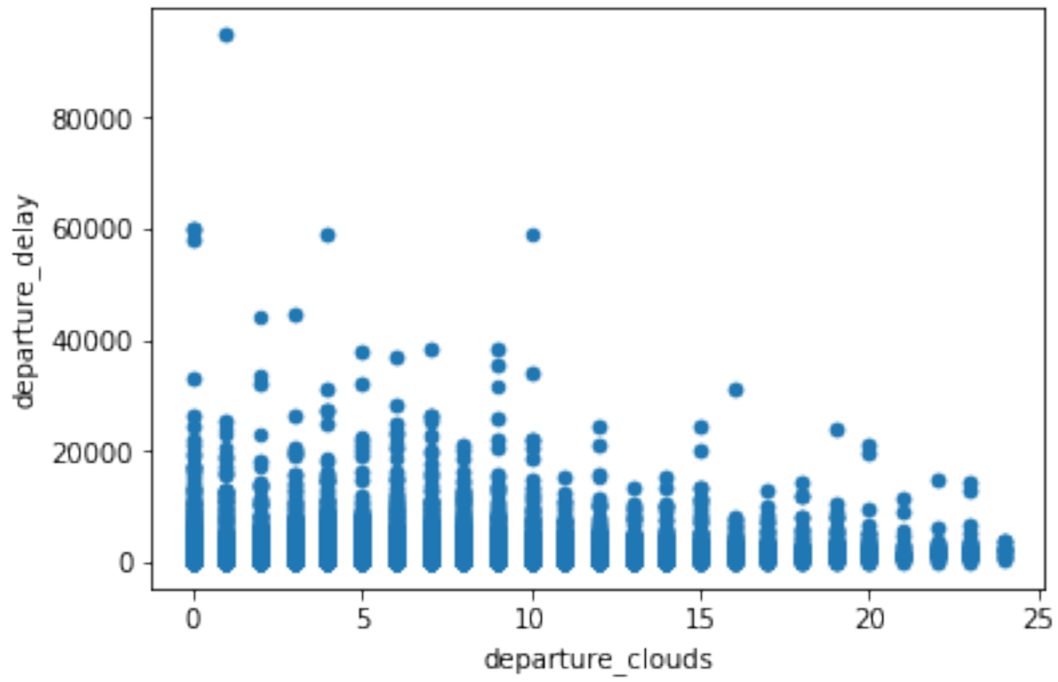
```
0.00012621855383365688
```

In [73]:

```
flight_weather.plot('departure_clouds', 'departure_delay', kind='scatter')
```

Out[73]:

<matplotlib.axes._subplots.AxesSubplot at 0x1203a7320>

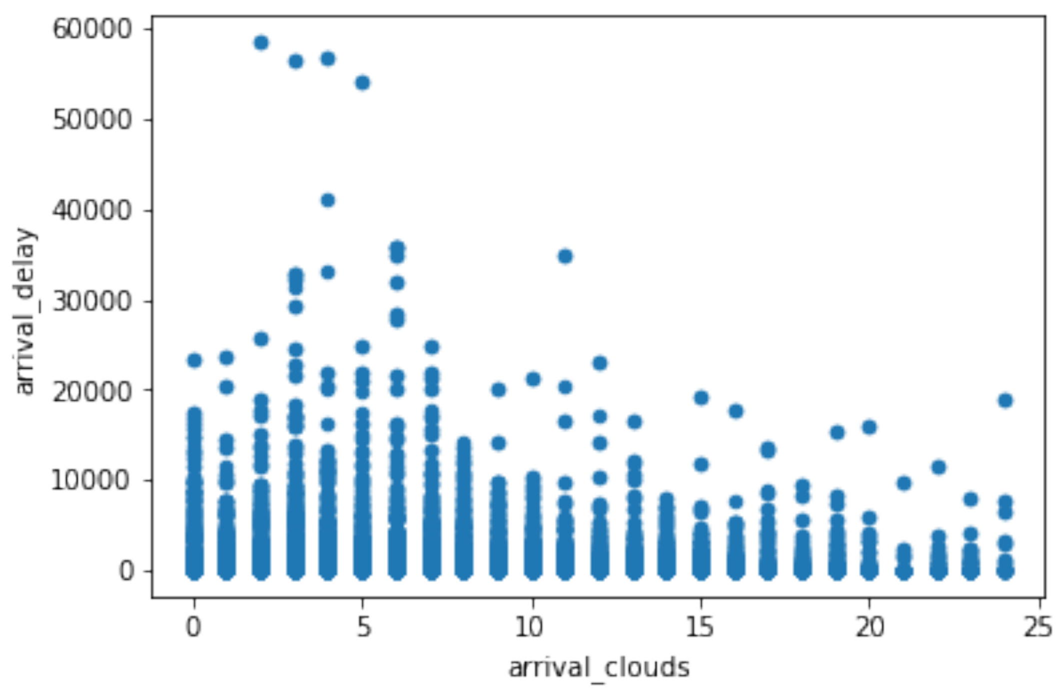


In [74]:

```
flight_weather.plot('arrival_clouds', 'arrival_delay', kind='scatter')
```

Out[74]:

<matplotlib.axes._subplots.AxesSubplot at 0x120482b70>

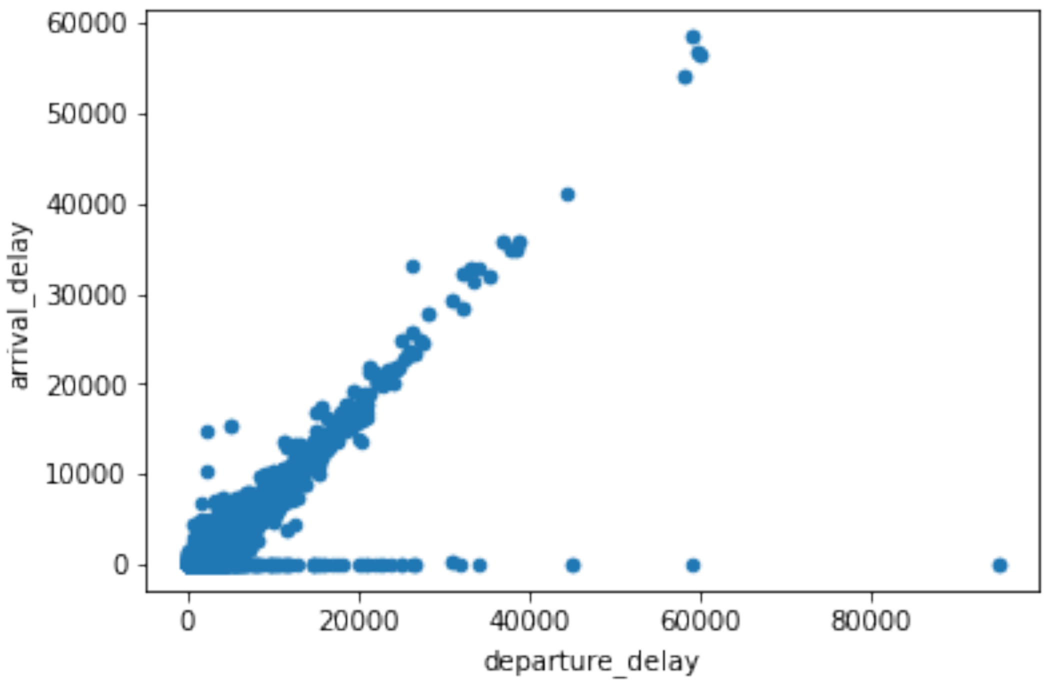


In [75]:

```
flight_weather.plot('departure_delay', 'arrival_delay', kind='scatter')
```

Out[75]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1207cb518>
```



In [76]:

```
print (explained_variance_score (flight_weather['arrival_delay'],flight_weathe
r['departure_delay']))
```

```
0.6200186730704473
```

In [77]:

```
print (r2_score (flight_weather['arrival_delay'],flight_weather['departure_del
ay']))
```

```
0.34271916634105437
```

Derive time-related features for recent delays and cancellations

In [78]:

```
flight_weather['departure_previous_hour'] = flight_weather['scheduled_departur
e_hour'].map(lambda h: h-1)
flight_weather['departure_day'] = flight_weather['departure_previous_hour'].ma
p(lambda h: np.rint((h)/24)).astype(int)
flight_weather['arrival_previous_hour'] = flight_weather['scheduled_arrival_ho
ur'].map(lambda h: h-1)
flight_weather['arrival_day'] = flight_weather['arrival_previous_hour'].map(la
mbda h: np.rint((h)/24)).astype(int)
flight_weather.dtypes
```

Out[78]:

```
mode                                    object

flight_number                           object
callsign                                object
aircraft_model_code                     object
aircraft_model_description              object
aircraft_registration                   object
airline_name                            object
airline_iata                            object
airline_icao                            object
flight_origin_code_iata                 object
flight_origin_code_icao                 object
flight_origin_name                      object
flight_origin_time_offset                int64
flight_destination_code_iata            object
flight_destination_code_icao            object
flight_destination_name                 object
flight_destination_time_offset           int64
flight_departure_scheduled               int64
flight_departure_real                  float64
flight_arrival_scheduled                 int64
flight_arrival_real                    float64
flight_duration                        float64
assumed_cancellation                     int64
scheduled_arrival_hour                   int64
scheduled_departure_hour                 int64
flight_duration_scheduled                int64
arrival_delay                          float64
departure_delay                        float64
origin_departure_delay                 float64
origin_cancellation                    float64
                                         ...
airline_cancellation                   float64
aircraft_departure_delay               float64
aircraft_arrival_delay                 float64
aircraft_cancellation                  float64
AIRPORT_IATA_destination                object
hours_destination                        int64
departure_METAR                         object
departure_weather                       object
AIRPORT_IATA_arrival                    object
hours_arrival                            int64
arrival_METAR                           object
arrival_weather                         object
departure_visibility                     int64
arrival_visibility                       int64
departure_temperature                    int64
arrival_temperature                      int64
freezing_at_departure                    int64
freezing_at_arrival                      int64
departure_windspeed                      int64
arrival_windspeed                        int64
departure_pressure                       int64
arrival_pressure                         int64
departure_dewpoint                       int64
arrival_dewpoint                         int64
departure_clouds                         int64
arrival_clouds                           int64
```

```
departure_previous_hour                int64
departure_day                          int64
arrival_previous_hour                  int64
arrival_day                            int64
Length: 63, dtype: object
```

In [79]:

```
departure_backlog_daily = flight_weather.groupby(['flight_origin_code_iata',
                              'departure_day']).agg({'departure_delay':'mean',
                                                     'assumed_cancellation':'mean'}
).reset_index().rename(
    columns={'departure_delay':'departure_backlog_today',
             'assumed_cancellation':'cancellations_today'})

departure_backlog_daily.head()
```

Out[79]:

	flight_origin_code_iata	departure_day	departure_backlog_today	cancellations_today
0	ATH	18035	2490.727273	0.0
1	ATH	18036	2078.193548	0.0
2	ATH	18037	1606.814815	0.0
3	ATH	18038	1404.600000	0.0
4	ATH	18039	1092.244444	0.0

In [80]:

```
departure_backlog = flight_weather.groupby(['flight_origin_code_iata',
                              'scheduled_departure_hour']).agg({'departure_delay
':'mean',
                                                     'assumed_cancellation':'mean'}
).reset_index().rename(
    columns={'departure_delay':'departure_backlog',
             'assumed_cancellation':'previous_cancellations'})

departure_backlog.head()
```

Out[80]:

	flight_origin_code_iata	scheduled_departure_hour	departure_backlog	previous_cancella
0	ATH	432849	3118.0	
1	ATH	432850	1269.5	
2	ATH	432851	2199.0	
3	ATH	432852	2895.0	
4	ATH	432853	1447.0	

In [81]:

```
features = pd.merge(flight_weather,departure_backlog_daily,
                          on=['flight_origin_code_iata','departure_day'],
                        suffixes=('_flights','_backlog'))

features = pd.merge(features,departure_backlog,
                          left_on=['flight_origin_code_iata','departure_previo
us_hour'],
                          right_on=['flight_origin_code_iata','scheduled_depar
ture_hour'],
                          suffixes=('_current','_previous'))

features.dtypes
```

Out[81]:

```
mode                                object
flight_number                       object
callsign                            object
aircraft_model_code                 object
aircraft_model_description          object
aircraft_registration               object
airline_name                        object
airline_iata                        object
airline_icao                        object
flight_origin_code_iata             object
flight_origin_code_icao             object
flight_origin_name                  object
flight_origin_time_offset            int64
flight_destination_code_iata        object
flight_destination_code_icao        object
flight_destination_name             object
flight_destination_time_offset       int64
flight_departure_scheduled           int64
flight_departure_real              float64
flight_arrival_scheduled             int64
flight_arrival_real                float64
flight_duration                    float64
assumed_cancellation                 int64
scheduled_arrival_hour               int64
scheduled_departure_hour_current     int64
flight_duration_scheduled            int64
arrival_delay                      float64
departure_delay                    float64
origin_departure_delay             float64
origin_cancellation                float64
                                    ...
hours_destination                    int64
departure_METAR                     object
departure_weather                   object
AIRPORT_IATA_arrival                object
hours_arrival                        int64
arrival_METAR                       object
arrival_weather                     object
departure_visibility                 int64
```

```
arrival_visibility                       int64
departure_temperature                    int64
arrival_temperature                      int64
freezing_at_departure                    int64
freezing_at_arrival                      int64
departure_windspeed                      int64
arrival_windspeed                        int64
departure_pressure                       int64
arrival_pressure                         int64
departure_dewpoint                       int64
arrival_dewpoint                         int64
departure_clouds                         int64
arrival_clouds                           int64
departure_previous_hour                  int64
departure_day                            int64
arrival_previous_hour                    int64
arrival_day                              int64
departure_backlog_today                float64
cancellations_today                    float64
scheduled_departure_hour_previous        int64
departure_backlog                      float64
previous_cancellations                 float64
Length: 68, dtype: object
```

In [82]:

```
features.head()
```

Out[82]:

	mode	flight_number	callsign	aircraft_model_code	aircraft_model_description	ai
0	arrivals	UA900	UAL900	B77W	Boeing 777-322(ER)	
1	departures	UA900	UAL900	B77W	Boeing 777-322(ER)	
2	arrivals	AA101	AAL101	B772	Boeing 777-223(ER)	
3	departures	AA101	AAL101	B772	Boeing 777-223(ER)	
4	arrivals	BA283	BAW283	B744	Boeing 747-436	

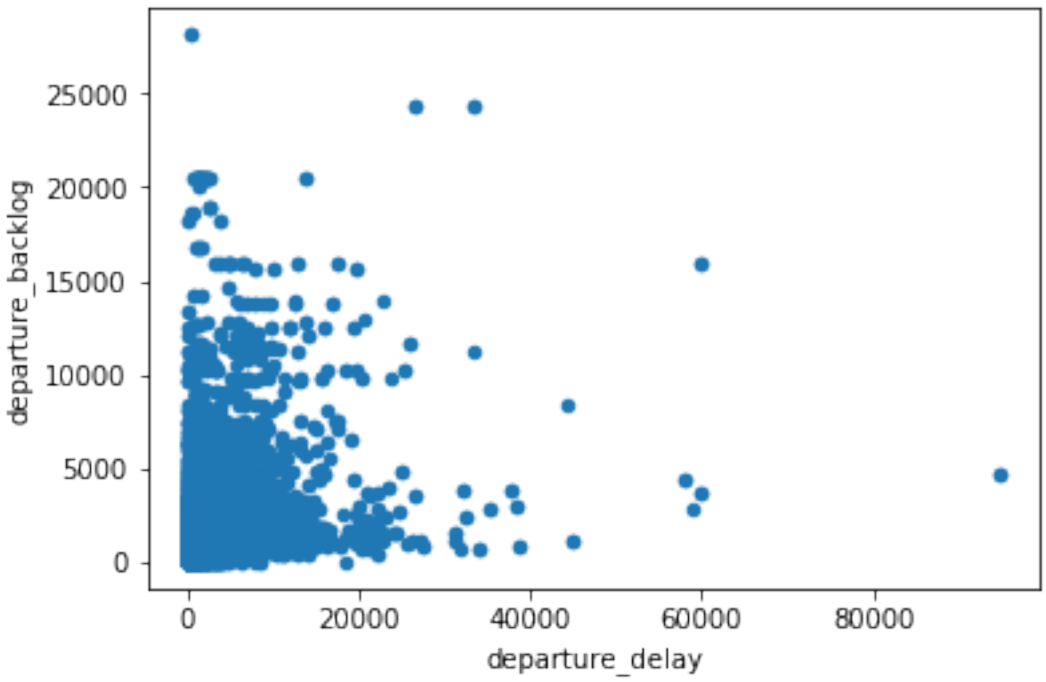
5 rows × 68 columns

In [83]:

```
features.plot('departure_delay','departure_backlog',kind='scatter')
```

Out[83]:

<matplotlib.axes._subplots.AxesSubplot at 0x120873320>

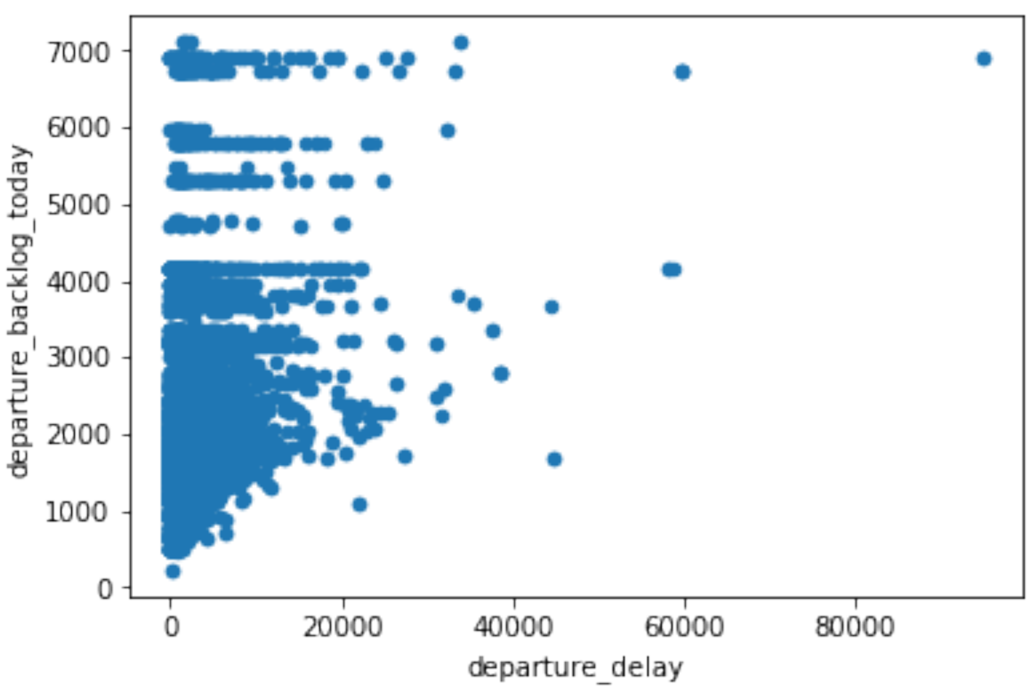


In [84]:

```
features.plot('departure_delay','departure_backlog_today',kind='scatter')
```

Out[84]:

<matplotlib.axes._subplots.AxesSubplot at 0x120b6c438>



In [85]:

```
arrival_backlog_daily = flight_weather.groupby(['flight_destination_code_iata'
,
                                               'arrival_day']).agg({'arrival_delay':'mean'}).rese
t_index().rename(
    columns={'arrival_delay':'arrival_backlog_today'})

arrival_backlog_daily.head()
```

Out[85]:

	flight_destination_code_iata	arrival_day	arrival_backlog_today
0	ATH	18035	1000.250000
1	ATH	18036	1526.490566
2	ATH	18037	599.535714
3	ATH	18038	492.016949
4	ATH	18039	561.784314

In [86]:

```
arrival_backlog = flight_weather.groupby(['flight_destination_code_iata',
                                          'scheduled_arrival_hour']).agg({'departure_delay':
'mean',}).reset_index().rename(
    columns={'departure_delay':'arrival_backlog'})

arrival_backlog.head()
```

Out[86]:

	flight_destination_code_iata	scheduled_arrival_hour	arrival_backlog
0	ATH	432851	1959.333333
1	ATH	432852	1908.000000
2	ATH	432853	1330.333333
3	ATH	432854	2099.000000
4	ATH	432855	1219.000000

In [87]:

```
features = pd.merge(features,arrival_backlog_daily, on=['flight_destination_co
de_iata','arrival_day'],
                    suffixes=('_current','_previous'))

features = pd.merge(features,arrival_backlog,
                           left_on=['flight_destination_code_iata','arrival_pre
vious_hour'],
                           right_on=['flight_destination_code_iata','scheduled_
arrival_hour'],
                           suffixes=('_current','_previous'))

features.dtypes
```

Out[87]:

```
mode                                  object
flight_number                         object
callsign                              object
aircraft_model_code                   object
aircraft_model_description            object
aircraft_registration                 object
airline_name                          object
airline_iata                          object
airline_icao                          object
flight_origin_code_iata               object
flight_origin_code_icao               object
flight_origin_name                    object
flight_origin_time_offset              int64
flight_destination_code_iata          object
flight_destination_code_icao          object
flight_destination_name               object
flight_destination_time_offset         int64
flight_departure_scheduled             int64
flight_departure_real                float64
flight_arrival_scheduled               int64
flight_arrival_real                  float64
flight_duration                      float64
assumed_cancellation                   int64
scheduled_arrival_hour_current         int64
scheduled_departure_hour_current       int64
flight_duration_scheduled              int64
arrival_delay                        float64
departure_delay                      float64
origin_departure_delay               float64
origin_cancellation                  float64
                                      ...
AIRPORT_IATA_arrival                  object
hours_arrival                          int64
arrival_METAR                         object
arrival_weather                       object
departure_visibility                   int64
arrival_visibility                     int64
departure_temperature                  int64
arrival_temperature                    int64
freezing_at_departure                  int64
```

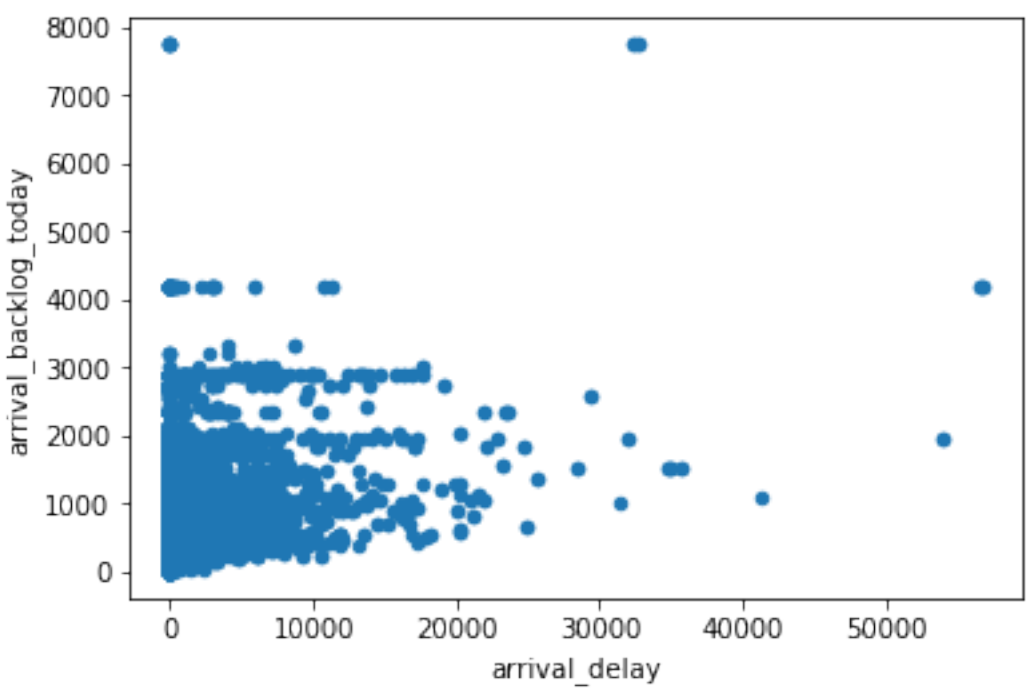
```
freezing_at_arrival                          int64
departure_windspeed                          int64
arrival_windspeed                            int64
departure_pressure                           int64
arrival_pressure                             int64
departure_dewpoint                           int64
arrival_dewpoint                             int64
departure_clouds                             int64
arrival_clouds                               int64
departure_previous_hour                      int64
departure_day                                int64
arrival_previous_hour                        int64
arrival_day                                  int64
departure_backlog_today                    float64
cancellations_today                        float64
scheduled_departure_hour_previous            int64
departure_backlog                          float64
previous_cancellations                     float64
arrival_backlog_today                      float64
scheduled_arrival_hour_previous              int64
arrival_backlog                            float64
Length: 71, dtype: object
```

In [88]:

```python
features.plot('arrival_delay','arrival_backlog_today',kind='scatter')
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e562748>
```



In [89]:

```python
print (explained_variance_score (features['arrival_delay'],features['arrival_backlog_today']))
```

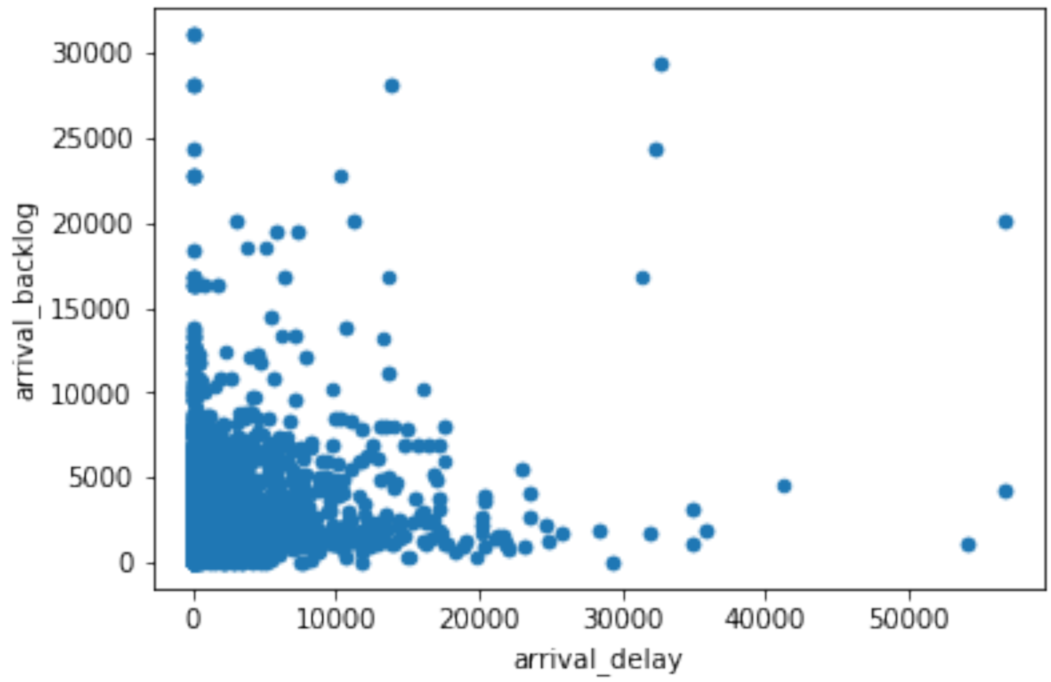
```
0.07122332574340884
```

In [90]:

```
features.plot('arrival_delay','arrival_backlog',kind='scatter')
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x120b090f0>
```



Summary so far: we derived founds lots of *weakly* correlated features for delays and two *strongly* correlated features for cancellations; there is also a strong correlation between departure delays and arrival delays, but apart from that, no single feature is significant *by itself*, so we need to find one or more models that use the available features to predict *departure* delays, cancellations and *arrival* delays.

Evaluate Linear Regression model for prediction of departure delays

In [91]:

```
from sklearn.linear_model import LinearRegression
```

In [92]:

```
from sklearn.model_selection import train_test_split
```

In [93]:

```
feature_labels = features[['departure_delay','assumed_cancellation','arrival_d
elay']]
```

In [94]:

```
feature_set = features[['departure_backlog','departure_backlog_today','departu
re_windspeed',
                                        'departure_visibility','departure_wi
ndspeed','departure_temperature',
                                        'departure_pressure','departure_dewpo
int','departure_clouds',
                                        'cancellations_today','previous_cance
llations',
                           'arrival_backlog','arrival_backlog_today','origin_d
eparture_delay',
                           'origin_cancellation','destination_arrival_delay',
                           'airline_departure_delay', 'airline_cancellation',
'airline_arrival_delay',
                           'aircraft_departure_delay', 'aircraft_cancellation'
, 'aircraft_arrival_delay',
                           'freezing_at_departure', 'freezing_at_arrival']]
```

In [96]:

```
from sklearn import preprocessing

scaled_features = preprocessing.scale (feature_set)
```

In [99]:

```
train_set, test_set, train_labels, test_labels = train_test_split(scaled_featu
res, feature_labels, test_size=0.7)
```

In [100]:

```
train_labels_departure_delay = train_labels[['departure_delay']]
test_labels_departure_delay = test_labels[['departure_delay']]
train_labels_departure_delay.head()
```

Out[100]:

	departure_delay
12797	3521.0
8679	9782.0
14395	3504.0
14691	3143.0
8812	2628.0

In [101]:

```
linreg = LinearRegression().fit(train_set, train_labels_departure_delay)
```

In [102]:

```
predicted_delay = linreg.predict(test_set)
```

In [103]:

```
predicted_delay.shape
```

Out[103]:

```
(14152, 1)
```

In [104]:

```
print (predicted_delay)
```

```
[[1900.4988819]
 [1352.4988819]
 [3792.4988819]
 ...
 [2472.4988819]
 [1432.4988819]
 [4148.4988819]]
```

In [105]:

```
actual_delay = np.array(test_labels_departure_delay)

print (actual_delay)
```

```
[[ 624.]
 [   0.]
 [4589.]
 ...
 [ 817.]
 [2249.]
 [2437.]]
```

In [106]:

```
actual_delay.shape
```

Out[106]:

```
(14152, 1)
```

In [107]:

```
print (r2_score (actual_delay, predicted_delay))
```

```
0.13847240873469013
```

In [108]:

```
print (explained_variance_score (actual_delay, predicted_delay))
```

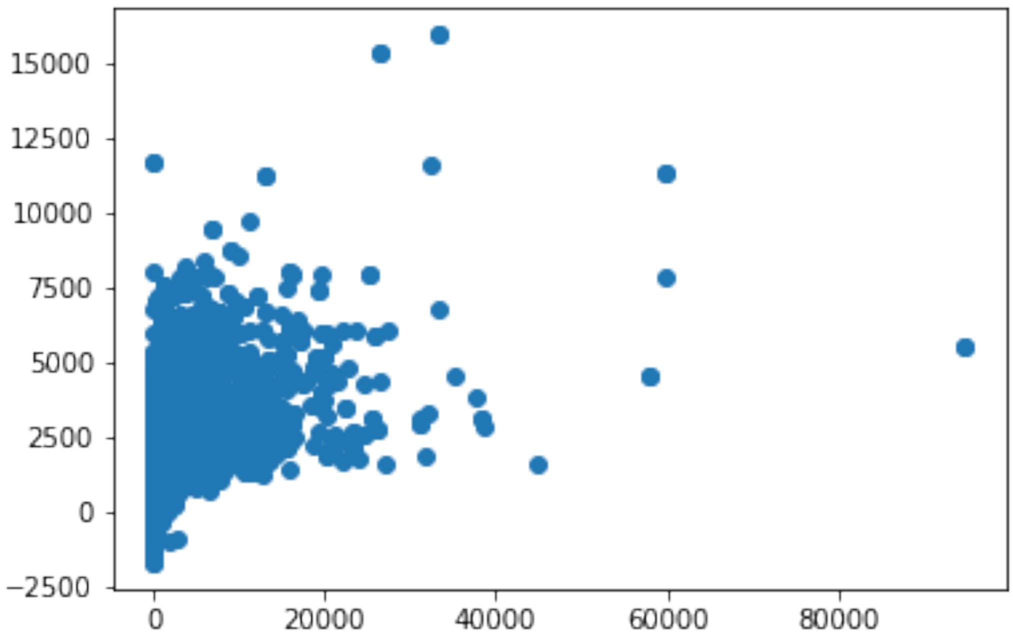
```
0.13847299985421124
```

In [109]:

```
plt.scatter(actual_delay,predicted_delay)
```

Out[109]:

```
<matplotlib.collections.PathCollection at 0x12107c828>
```



Evaluate Random Forest model for prediction of departure delays

In [110]:

```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(random_state=4, n_estimators=700)

rf.fit(train_set, np.ravel(train_labels_departure_delay))
```

Out[110]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=N
one,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_spli
t=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=7
00,
                      n_jobs=None, oob_score=False, random_state=4
, verbose=0,
                      warm_start=False)
```

In [113]:

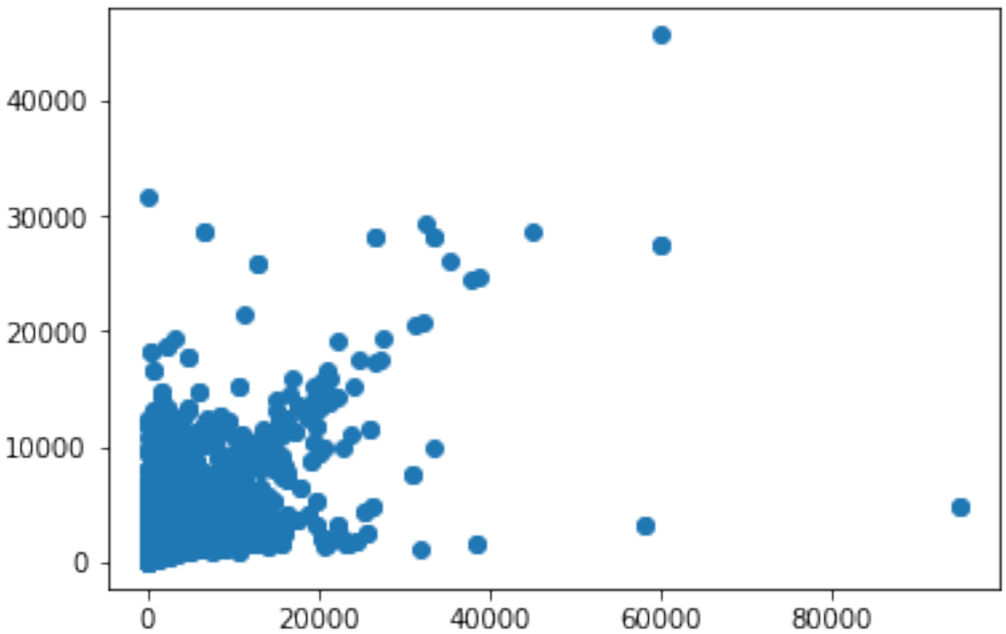
```
rf_predicted_delay = rf.predict(test_set)
```

In [114]:

```
plt.scatter(actual_delay,rf_predicted_delay)
```

Out[114]:

<matplotlib.collections.PathCollection at 0x12b1f4c18>



In [115]:

```
print (r2_score (actual_delay, rf_predicted_delay))
```

0.3087056466369782

In [116]:

```
print (explained_variance_score (actual_delay, rf_predicted_delay))
```

0.3115547321963949 6

Evaluate Gradient Booster model for prediction of departure delays

In [117]:

```
from sklearn.ensemble import GradientBoostingRegressor

gb = GradientBoostingRegressor(random_state=4, n_estimators=700)

gb.fit(train_set, np.ravel(train_labels_departure_delay))
```

Out[117]:

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', ini
t=None,
                          learning_rate=0.1, loss='ls', max_depth=
3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_
split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimato
rs=700,
                          n_iter_no_change=None, presort='auto', r
andom_state=4,
                          subsample=1.0, tol=0.0001, validation_fr
action=0.1,
                          verbose=0, warm_start=False)
```

In [118]:

```
gb_predicted_delay = gb.predict(test_set)
```

In [119]:

```
print (explained_variance_score (actual_delay, gb_predicted_delay))
```

```
0.22161936917622538
```

In [120]:

```
print (r2_score (actual_delay, gb_predicted_delay))
```

```
0.22131543840515155
```

Evaluate Support Vector Machine model for prediction of cancellations

In [121]:

```
train_labels_cancellation = train_labels[['assumed_cancellation']]
test_labels_cancellation = test_labels[['assumed_cancellation']]
train_labels_cancellation.head()
```

Out[121]:

	assumed_cancellation
12797	0
8679	0
14395	0
14691	0
8812	0

In [122]:

```
from sklearn import svm
```

In [123]:

```
svc = svm.SVC(gamma='scale', class_weight='balanced') # automatically give hig
her weight to under-represented classes
```

In [124]:

```
svc.fit(train_set,np.ravel(train_labels_cancellation))
```

Out[124]:

```
SVC(C=1.0, cache_size=200, class_weight='balanced', coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel
='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=T
rue,
    tol=0.001, verbose=False)
```

In [125]:

```
predicted_cancellations = svc.predict(test_set)
```

In [126]:

```
actual_cancellations = np.ravel(test_labels_cancellation)
print (actual_cancellations)
```

```
[0 1 0 ... 0 0 0]
```

In [127]:

```
from sklearn.metrics import roc_auc_score

print (roc_auc_score (actual_cancellations, predicted_cancellations))
```

```
0.8645881178810089
```

An area-under-the-curve of 50% is no better than chance, so the balanced SVC is significantly better than chance

In [128]:

```
from sklearn.metrics import confusion_matrix

print (confusion_matrix (actual_cancellations, predicted_cancellations))
```

```
[[12953   738]
 [  100   361]]
```

In [129]:

```
from sklearn.metrics import balanced_accuracy_score

print (balanced_accuracy_score (actual_cancellations, predicted_cancellations)
)
```

```
0.8645881178810089
```

Evaluate Stochastic Gradient Descent model for prediction of cancellations

In [130]:

```
from sklearn.linear_model import SGDClassifier
```

In [131]:

```
sgc = SGDClassifier(loss="log", penalty="l2", max_iter=50000, early_stopping=T
rue, class_weight='balanced')
```

In [132]:

```
sgc.fit(train_set,np.ravel(train_labels_cancellation))
```

Out[132]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight='balanced'
,
              early_stopping=True, epsilon=0.1, eta0=0.0, fit_inte
rcept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='log',
              max_iter=50000, n_iter_no_change=5, n_jobs=None, pen
alty='l2',
              power_t=0.5, random_state=None, shuffle=True, tol=0.
001,
              validation_fraction=0.1, verbose=0, warm_start=False
)
```

In [133]:

```
sgc_predicted_cancellations = sgc.predict(test_set)
```

In [134]:

```
print (roc_auc_score (actual_cancellations, sgc_predicted_cancellations))
```

```
0.8119114461722642
```

In [135]:

```
print (balanced_accuracy_score (actual_cancellations, sgc_predicted_cancellati
ons))
```

```
0.8119114461722641
```

In [136]:

```
print (confusion_matrix (actual_cancellations, sgc_predicted_cancellations))
```

```
[[9699 3992]
 [  39  422]]
```

In [137]:

```
from sklearn.ensemble import VotingClassifier
```

Evaluate Linear Regression model for prediction of arrival delays

In [138]:

```
train_labels_arrival_delay = train_labels[['arrival_delay']]
test_labels_arrival_delay = test_labels[['arrival_delay']]
train_labels_arrival_delay.head()
```

Out[138]:

	arrival_delay
12797	1945.0
8679	7275.0
14395	1769.0
14691	49.0
8812	1148.0

In [139]:

```
linreg_arrivals = LinearRegression().fit(train_set, train_labels_arrival_delay
)
```

In [140]:

```
predicted_arrival_delay = linreg_arrivals.predict(test_set)
```

In [141]:

```
actual_arrival_delay = np.array(test_labels_arrival_delay)
```

In [142]:

```
print (r2_score (actual_arrival_delay, predicted_arrival_delay))
```

```
0.11890286211010781
```

In [143]:

```
print (explained_variance_score (actual_arrival_delay, predicted_arrival_delay
))
```

```
0.11896392882094797
```

Evaluate Support Vector Machine model for prediction of arrival delays

In [144]:

```
from sklearn.svm import SVR
```

In [145]:

```
svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
```

In [147]:

```
svr_rbf.fit(train_set, np.ravel(train_labels_arrival_delay))
```

Out[147]:

```
SVR(C=100, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma
=0.1,
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=
False)
```

In [148]:

```
pred_rbf = svr_rbf.predict(test_set)
```

In [149]:

```
print (r2_score (actual_arrival_delay, pred_rbf))
```

```
-0.05395238411185499
```

In [150]:

```
print (explained_variance_score (actual_arrival_delay, pred_rbf))
```

```
0.023640468842816675
```

In [ ]:

```
## _TO DO:_ More investigation of the data and models for arrival and departur
e delays
```

In [ ]:

```

```