Exploring data aviation

Python notebook utilisant les données de la base de données et synopsis des accidents aériens · 1,970 vues · Il y a 3 ans

Version 6
6 commit

Exporing aviation accidents data

Link to kagge: https://www.kaggle.com/khsamaha/aviation-accident-database-synopses (https://www.kaggle.com/khsamaha/aviation-accident-database-synopses)

Dataset purpose

In some comment I've read interesting questions about this dataset:

- Which is the type accident often to happen? Which are the features relevant?
- What is season that there are more accident?
- The amateur have a influence on accident or injury severity?
- Do they take too long to make preliminary reports?
- What do scheme have more accident?
- Where are there more accident? - deprecated
- What do aircraft have more accident? -deprecated
- How do accidents evolve in the time of aviation in the United States?

Credits:

- I took some useful functions from https://www.kaggle.com/helgejo/titanic/an-interactive-data-science-tutorial (https://www.kaggle.com/helgejo/titanic/an-interactive-data-science-tutorial)

NOTE this is a work in progress

In [1]:

```
# Python libraries
import math
import re
import datetime


# Handle table-like data and matrices
import numpy as np
```



```
# Modelling Algorithms
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier , GradientBoosting
Classifier

# Modelling Helpers
from sklearn.preprocessing import StandardScaler, Imputer , Normalizer
, scale
from sklearn.cross_validation import train_test_split , StratifiedKFol
d
from sklearn.feature_selection import RFECV

# Visualisation
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import seaborn as sns
```

```
# Configure visualisations
%matplotlib inline
mpl.style.use( 'ggplot' )
sns.set_style( 'white' )
pylab.rcParams[ 'figure.figsize' ] = 12 , 10
```

```
/opt/conda/lib/python3.5/site-packages/sklearn/cross_validation.py:44:
DeprecationWarning: This module was deprecated in version 0.18 in favo
r of the model_selection module into which all the refactored classes
and functions are moved. Also note that the interface of the new CV it
erators are different from that of this module. This module will be re
moved in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Plot and data study helpers

In [2]:

```
def plot_histograms( df , variables , n_rows , n_cols ):
    fig = plt.figure( figsize = ( 16 , 12 ) )
    for i, var_name in enumerate( variables ):
        ax=fig.add_subplot( n_rows , n_cols , i+1 )
        df[ var_name ].hist( bins=10 , ax=ax )
        ax.set_title( 'Skew: ' + str( round( float( df[ var_name ].ske
w() ) , ) ) ) # + ' ' + var_name ) #var_name+" Distribution")
        ax.set_xticklabels( [] , visible=False )
        ax.set_yticklabels( [] , visible=False )
    fig.tight_layout()  # Improves appearance a bit.
    plt.show()

def plot_distribution( df , var , target , **kwargs ):
    row = kwargs.get( 'row' , None )
    col = kwargs.get( 'col' , None )
    facet = sns.FacetGrid( df , hue=target , aspect=4 , row = row , co
l = col )
    facet.map( sns.kdeplot , var , shade= True )
    facet.set( xlim=( df[ var ].min() , df[ var ].max() ) )
    facet.add_legend()

def plot_categories( df , cat , target , **kwargs ):
    row = kwargs.get( 'row' , None )
    col = kwargs.get( 'col' , None )
    facet = sns.FacetGrid( df , row = row , col = col )
    facet.map( sns.barplot , cat , target )
    facet.add_legend()

def plot_correlation_map( df ):
    corr = df.corr()
    _ , ax = plt.subplots( figsize =( 12 , 10 ) )
    cmap = sns.diverging_palette( 220 , 10 , as_cmap = True )
    _ = sns.heatmap(
        corr,
        cmap = cmap,
        square=True,
        cbar_kws={ 'shrink' : .9 },
        ax=ax,
        annot = True,
        annot_kws = { 'fontsize' : 12 }
    )

def describe_more( df ):
```

```
    var = [] ; l = [] ; t = []
    for x in df:
        var.append( x )
        l.append( len( pd.value_counts( df[ x ] ) ) )
        t.append( df[ x ].dtypes )
    levels = pd.DataFrame( { 'Variable' : var , 'Levels' : l , 'Dataty
pe' : t } )
    levels.sort_values( by = 'Levels' , inplace = True )
    return levels

def plot_variable_importance( X , y ):
    tree = DecisionTreeClassifier( random_state = 99 )
    tree.fit( X , y )
    plot_model_var_imp( tree , X , y )

def plot_model_var_imp( model , X , y ):
    imp = pd.DataFrame(
        model.feature_importances_  ,
        columns = [ 'Importance' ] ,
        index = X.columns
    )
    imp = imp.sort_values( [ 'Importance' ] , ascending = True )
    imp[ : 10 ].plot( kind = 'barh' )
    print (model.score( X , y ))

def category_values(dataframe, categories):
    for c in categories:
        print('\n', dataframe.groupby(by=c)[c].count().sort_values(asc
ending=False))
        print('Nulls: ', dataframe[c].isnull().sum())
```

Loading data

In [3]:

```
df = pd.read_csv('../input/AviationDataEnd2016UP.csv', sep=',', header
=0, encoding = 'iso-8859-1')

df.sample(10)
```

Out[3]:

	Event.Id	Investigation.Type	Accident.Number	Event.Date	Location	Countr
4937	20130909X44026	Accident	CEN13LA537	2013-09-06	Arlington, MN	United States
77628	20020917X03748	Accident	MIA82DA139	1982-06-21	NEAR HORSESHOE, FL	United States
36441	20001208X09006	Accident	FTW98LA009B	1997-10-10	ALBUQUERQUE, NM	United States
34541	20001211X10912	Accident	MIA98FA229	1998-08-23	CARROLLTON, AL	United States
25922	20020909X01554	Incident	MIA02IA160	2002-08-26	Bradenton, FL	United States
54826	20001213X29750	Accident	BFO90LA011	1989-11-14	WILMINGTON, DE	United States
18319	20060727X01026	Accident	DEN06CA092	2006-07-04	Jackson, WY	United States
122	20161024X11610	Accident	CEN17LA025	2016-10-23	Buffalo, WY	United States

14063	20080917X01486	Accident	LAX08LA271	2008-08-20	Reno, NV	United States
41199	20001207X04117	Accident	ANC95FA157	1995-08-18	KAKTOVIK, AK	United States

10 rows × 31 columns

Getting info on the fields types

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79293 entries, 0 to 79292
Data columns (total 31 columns):
Event.Id                  79293 non-null object
Investigation.Type        79293 non-null object
Accident.Number           79293 non-null object
Event.Date                79293 non-null object
Location                  79215 non-null object
Country                   78786 non-null object
Latitude                  25751 non-null float64
Longitude                 25742 non-null float64
Airport.Code              44666 non-null object
Airport.Name              47439 non-null object
Injury.Severity           79293 non-null object
Aircraft.Damage           76883 non-null object
Aircraft.Category         22477 non-null object
Registration.Number       76209 non-null object
Make                      79204 non-null object
Model                     79175 non-null object
Amateur.Built             78721 non-null object
Number.of.Engines         75175 non-null float64
Engine.Type               75919 non-null object
FAR.Description           22334 non-null object
Schedule                  11501 non-null object
Purpose.of.Flight         75399 non-null object
Air.Carrier               3918 non-null object
Total.Fatal.Injuries      55984 non-null float64
Total.Serious.Injuries    53742 non-null float64
Total.Minor.Injuries      54833 non-null float64
Total.Uninjured           66949 non-null float64
Weather.Condition         77136 non-null object
Broad.Phase.of.Flight     73239 non-null object
Report.Status             79293 non-null object
Publication.Date          65819 non-null object
dtypes: float64(7), object(24)
memory usage: 18.8+ MB
```

Let's see what kind of numeric data we have

In [5]:

```
df.describe()
```

Out[5]:

	Latitude	Longitude	Number.of.Engines	Total.Fatal.Injuries	Total.Serious.Injuries	To

count	25751.000000	25742.000000	75175.000000	55984.000000	53742.000000	54
mean	37.690421	-93.781061	1.148055	0.814679	0.317703	0.5
std	12.148019	39.243662	0.453847	6.233700	1.372924	2.7
min	-78.016945	-178.676111	0.000000	0.000000	0.000000	0.0
25%	33.379445	-115.008542	1.000000	0.000000	0.000000	0.0
50%	38.184166	-94.498055	1.000000	0.000000	0.000000	0.0
75%	42.566528	-81.725834	1.000000	1.000000	0.000000	1.0
max	89.218056	177.557778	18.000000	349.000000	111.000000	38

Getting some counts on how many different values are there for each feature

In [6]:
```
describe_more(df)
```

Out[6]:

	Datatype	Levels	Variable
1	object	2	Investigation.Type
16	object	2	Amateur.Built
27	object	3	Weather.Condition
20	object	3	Schedule
11	object	3	Aircraft.Damage
29	object	4	Report.Status
17	float64	6	Number.of.Engines
28	object	12	Broad.Phase.of.Flight
12	object	13	Aircraft.Category
18	object	14	Engine.Type
19	object	17	FAR.Description
21	object	22	Purpose.of.Flight
24	float64	40	Total.Serious.Injuries
25	float64	62	Total.Minor.Injuries
23	float64	122	Total.Fatal.Injuries
10	object	124	Injury.Severity
5	object	177	Country
26	float64	364	Total.Uninjured
22	object	2866	Air.Carrier
30	object	3591	Publication.Date
14	object	7475	Make
8	object	9631	Airport.Code
15	object	11330	Model
3	object	12638	Event.Date
6	float64	17665	Latitude
7	float64	18925	Longitude
9	object	22761	Airport.Name
4	object	25264	Location
13	object	68960	Registration.Number
0	object	78143	Event.Id
2	object	79293	Accident.Number

2	object	79293	Accident.Number

In [7]:

```
# splitting date field in the components

df['Year'] = df['Event.Date'].apply(lambda d: datetime.datetime.strptime(d, "%Y-%m-%d").year)
df['Month'] = df['Event.Date'].apply(lambda d: datetime.datetime.strptime(d, "%Y-%m-%d").month)
df['Day'] = df['Event.Date'].apply(lambda d: datetime.datetime.strptime(d, "%Y-%m-%d").day)

df = df[df['Year'] >= 1982]
```

Looking at some categories

I try to list some unique values in the categories fields to subsequently plot some data distribution over those.

In [8]:

```
categories = ['Investigation.Type',
              'Aircraft.Damage',
              'Aircraft.Category',
              'Amateur.Built',
              'Number.of.Engines',
              'Engine.Type',
              'FAR.Description',
              'Schedule',
              'Purpose.of.Flight',
              'Weather.Condition',
              'Broad.Phase.of.Flight',
              'Report.Status',
              'Air.Carrier']

for c in categories:
    print(c , df[c].unique())
```

```
Investigation.Type ['Accident' 'Incident']
Aircraft.Damage ['Substantial' 'Destroyed' nan 'Minor']
Aircraft.Category ['Airplane' 'Helicopter' 'Weight-Shift' 'Glider' 'Unknown' 'Balloon'
 'Powered Parachute' 'Ultralight' 'Gyroplane' 'Gyrocraft' nan
 'Powered-Lift' 'Rocket' 'Blimp']
Amateur.Built ['Yes' 'No' nan]
Number.of.Engines [ nan   1.   2.   0.   4.   3.  18.]
Engine.Type ['Reciprocating' nan 'Turbo Prop' 'Turbo Fan' 'Turbo Shaft' 'Unknown'
 'Turbo Jet' 'Electric' 'REC, ELEC' 'None' 'TF, TJ' 'Hybrid Rocket'
 'REC, TJ, TJ' 'REC, TJ, REC, TJ' 'TJ, REC, REC, TJ']
FAR.Description ['Part 91: General Aviation' nan 'Part 135: Air Taxi & Commuter'
 'Public Aircraft' 'Part 121: Air Carrier' 'Unknown'
 'Non-U.S., Non-Commercial' 'Part 137: Agricultural' 'Non-U.S., Commercial'
 'Part 103: Ultralight' 'Part 133: Rotorcraft Ext. Load' 'Public Use'
 'Part 129: Foreign' 'Armed Forces' 'Part 437: Commercial Space Flight'
 'Part 91 Subpart K: Fractional' 'Part 125: 20+ Pax,6000+ lbs'
 'Part 91F: Special Flt Ops.']
Schedule [nan 'NSCH' 'SCHD' 'UNK']
```

```
Schedule [nan 'NSCH' 'SCHD' 'UNK']
Purpose.of.Flight ['Personal' nan 'Instructional' 'Public Aircraft - F
ederal'
 'Public Aircraft - Local' 'Business' 'Positioning' 'Aerial Observatio
n'
 'Unknown' 'Aerial Application' 'Public Aircraft - State' 'Ferry'
 'Flight Test' 'Air Race/Show' 'Other Work Use' 'Skydiving' 'External
Load'
 'Glider Tow' 'Air Drop' 'Banner Tow' 'Executive/Corporate' 'Firefight
ing'
 'Public Aircraft']
Weather.Condition ['VMC' 'IMC' nan 'UNK']
Broad.Phase.of.Flight ['CRUISE' nan 'LANDING' 'TAKEOFF' 'DESCENT' 'APP
ROACH' 'OTHER' 'TAXI'
 'GO-AROUND' 'MANEUVERING' 'UNKNOWN' 'STANDING' 'CLIMB']
Report.Status ['Preliminary' 'Foreign' 'Factual' 'Probable Cause']
Air.Carrier [nan 'Aerowest Aviation (DBA: Redtail Air)'
 'Key Lime Air (DBA: Key Lime Air)' ..., 'EXECUTIVE CHARTER SERVICE'
 'LANG AIR SERVICE' 'ROCKY MOUNTAIN HELICOPTERS, IN']
```

Counting the number of different values for each category feature

In [9]:

```
category_values(df, categories)
```

```
 Investigation.Type
Accident    76112
Incident     3175
Name: Investigation.Type, dtype: int64
Nulls:  0

 Aircraft.Damage
Substantial    57049
Destroyed      17316
Minor           2512
Name: Aircraft.Damage, dtype: int64
Nulls:  2410

 Aircraft.Category
Airplane            19273
Helicopter           2360
Glider                381
Balloon               175
Gyrocraft             100
Weight-Shift           66
Powered Parachute      48
Unknown                32
Ultralight             31
Powered-Lift            5
Blimp                   3
Gyroplane               2
Rocket                  1
Name: Aircraft.Category, dtype: int64
Nulls:  56810

 Amateur.Built
No     71099
Yes     7616
Name: Amateur.Built, dtype: int64
Nulls:  572
```

```
Nulls:  372

 Number.of.Engines
1.0     63077
2.0     10057
0.0      1143
3.0       477
4.0       415
18.0        1
Name: Number.of.Engines, dtype: int64
Nulls:  4117

 Engine.Type
Reciprocating       64593
Turbo Shaft          3305
Turbo Prop           3042
Turbo Fan            2226
Unknown              2052
Turbo Jet             678
None                    6
TF, TJ                  3
Electric                3
REC, TJ, TJ             2
TJ, REC, REC, TJ        1
REC, TJ, REC, TJ        1
REC, ELEC               1
Hybrid Rocket           1
Name: Engine.Type, dtype: int64
Nulls:  3373

 FAR.Description
Part 91: General Aviation           17958
Part 137: Agricultural               1104
Non-U.S., Non-Commercial              771
Part 135: Air Taxi & Commuter         763
Part 121: Air Carrier                 525
Non-U.S., Commercial                  514
Part 129: Foreign                     194
Unknown                               181
Public Use                            179
Part 133: Rotorcraft Ext. Load         96
Part 91 Subpart K: Fractional          13
Public Aircraft                        12
Part 103: Ultralight                    8
Part 125: 20+ Pax,6000+ lbs             7
Armed Forces                            7
Part 437: Commercial Space Flight       1
Part 91F: Special Flt Ops.              1
Name: FAR.Description, dtype: int64
Nulls:  56953

 Schedule
UNK     4099
NSCH    3866
SCHD    3536
Name: Schedule, dtype: int64
Nulls:  67786

 Purpose.of.Flight
Personal                    44544
Instructional                9487
Unknown                      6771
Aerial Application           4369
Business                     3868
```

```
Positioning                1507
Other Work Use             1121
Ferry                       775
Public Aircraft             707
Aerial Observation          673
Executive/Corporate         515
Flight Test                 316
Skydiving                   155
Air Race/Show               146
Public Aircraft - Federal    88
External Load                83
Banner Tow                   81
Public Aircraft - State      57
Public Aircraft - Local      55
Glider Tow                   43
Firefighting                 21
Air Drop                     11
Name: Purpose.of.Flight, dtype: int64
Nulls:  3894

 Weather.Condition
VMC    70506
IMC     5657
UNK      967
Name: Weather.Condition, dtype: int64
Nulls:  2157

 Broad.Phase.of.Flight
LANDING        19209
TAKEOFF        15284
CRUISE         10746
MANEUVERING     9818
APPROACH        7719
TAXI            2322
CLIMB           2279
DESCENT         2202
GO-AROUND       1608
STANDING        1219
UNKNOWN          670
OTHER            157
Name: Broad.Phase.of.Flight, dtype: int64
Nulls:  6054

 Report.Status
Probable Cause    73917
Foreign            3966
Preliminary        1090
Factual             314
Name: Report.Status, dtype: int64
Nulls:  0

 Air.Carrier
UNITED AIRLINES                                                       49
AMERICAN AIRLINES                                                     41
CONTINENTAL AIRLINES                                                  25
USAIR                                                                 24
SOUTHWEST AIRLINES CO                                                 24
DELTA AIR LINES INC                                                   24
AMERICAN AIRLINES, INC.                                               22
CONTINENTAL AIRLINES, INC.                                            19
AMERICAN AIRLINES INC                                                 17
UNITED AIR LINES INC                                                  15
Delta Air Lines                                                       13
```

```
US AIRWAYS INC                                          12
SIMMONS AIRLINES (DBA: AMERICAN EAGLE)                  12
United Airlines                                         11
TRANS WORLD AIRLINES                                    11
DELTA AIRLINES                                          11
NORTHWEST AIRLINES                                      10
(DBA: AMERICAN AIRLINES)                                10
DELTA AIR LINES                                         10
Southwest Airlines                                      10
(DBA: UNITED EXPRESS)                                   10
American Airlines                                       10
(DBA: [EMS])                                            10
EASTERN AIRLINES                                         9
AMERICA WEST AIRLINES, INC.                              8
DELTA AIRLINES, INC.                                     8
HORIZON AIR                                              8
AMERICA WEST AIRLINES                                    8
(DBA: PENAIR)                                            8
FEDERAL EXPRESS CORP                                     7
                                                        ..
MARK AIR, INC (DBA: MARK AIR)                            1
MARK AIR INC.                                            1
MARK AIR EXPRESS                                         1
MARITIME HELICOPTERS INC (DBA: Maritime Helicopter)      1
MARITIME HELICOPTERS                                     1
MARCO AVIATION, INC                                      1
MANUIWA AIRWAYS (DBA: VOLCANO HILI-TOURS)                1
MANUFACTURED HOMES OF ALASKA INC (DBA: Bear Lake Air)    1
MANOKOTAK AIRWAYS                                        1
MAUI AIRLINES                                            1
MAUNA KEA HELICOPTERS, INC.                              1
MAXAIR                                                   1
MED TRANS CORP                                           1
MESA AIRLINES                                            1
MESA AIR SHUTTLE                                         1
MESA AIR GROUP, INC. (DBA: AMERICA WEST)                 1
MERLIN EXPRESS                                           1
MERCY FLIGHT                                             1
MERCURY AIRCOURIER SERVICE                               1
MERCURY AIR COURIER SERVICE                              1
MCMAHAN GUIDE, FLYING SERVICE                            1
MAY AIR EXPRESS                                          1
MCCAULEY AIR CENTER                                      1
MCCALL AVIATION INC (DBA: McCall Air)                    1
MCBRIDE, MICHAEL S. (DBA: AIR ADVENTURES , INC.)         1
MC CAULLY AIR SERVICE                                    1
MBD CORPORATION                                          1
MAYO AVIAITON INC                                        1
MAYEUX'S FLYING SERVICE INC                              1
(DBA: 40 MILE AIR, LTD)                                  1
Name: Air.Carrier, dtype: int64
Nulls:  75369
```

Filling Null values

The data is full of Null values. I'll try to fix the nulls copying data from the rest of the dataset when possible. For the rest I'll put 'unknown' strings.

In [10]:

```
# null damages can't be defined
```

```
df[df['Aircraft.Damage'].isnull()]
df['Aircraft.Damage'].fillna('Unknown', inplace=True)

# Fixing phase of flight nulls
df['Broad.Phase.of.Flight'].fillna('UNKNOWN', inplace=True)

# Fixing weather conditions
df['Weather.Condition'].fillna('UNK', inplace=True)

# null categories can't be defined
df['Aircraft.Category'].fillna('Unknown', inplace=True)

# can't define purpose of flight
df['Purpose.of.Flight'].fillna('Unknown', inplace=True)

# don't know ho to set missing schedules
df['Schedule'].fillna('UNK', inplace=True)

# don't know ho to set missing FAR.Description
df['FAR.Description'].fillna('Unknown', inplace=True)

# don't know ho to set missing Aircraft.Damage
df['Aircraft.Damage'].fillna('Unknown', inplace=True)

# don't know ho to set missing Air Carriers
df['Air.Carrier'].fillna('Unknown', inplace=True)

# don't know ho to set missing Makers
df['Make'].fillna('UNKNOWN', inplace=True)

# don't know ho to set missing Models
df['Model'].fillna('Unknown', inplace=True)

# don't know ho to set missing airport names
df['Airport.Name'].fillna('Unknown', inplace=True)

# don't know ho to set missing Models
df['Airport.Code'].fillna('Unknown', inplace=True)

# don't know ho to set missing Locations
df['Location'].fillna('Unknown', inplace=True)
```

Amateur producers

Instead of putting an 'unknown' value in the Amateur.Built field, I've collected all the producers and all the amateurs brands/names from the rest of the dataset and filled the null cells searching in the resulting two lists. For the remaining marks that are not present anywhere in the dataset I chose to set them as amateurs.

In [11]:

```
# Extracting producers and amateurs
producers = [x for x in df['Make'][df['Amateur.Built']== 'No'].unique
() ]
amateurs  = [x for x in df['Make'][df['Amateur.Built']== 'Yes'].unique
() ]

# -----------------------------------------------
# Function that fixes the null in amateur.built
def fix_amateur_built(ab, m):
    if type(ab) == str:
        return ab
    else:
```

```
        if m in producers:
            return 'No'
        else:
            return 'Yes'
# Fix for Amateur.Built field
am_built = df.apply(lambda x: fix_amateur_built(x['Amateur.Built'], x[
'Make']), axis=1)
df = df.assign(AmateurBuilt = am_built, index=df.index)
```

Number of engines

For the balloons I'll set this value to 0. For the remaining, I'll make some assumptions and aproximations based on the rest of the values.

In [12]:

```
# Function that fixes the null in number.of.engines
def fix_number_of_engines(noe, m):
    if noe >= 0:
        return noe
    else:
        # Setting number of engines at the mean number of engines for t
he producer
        r = np.round(df['Number.of.Engines'][df['Make']==m].mean())
        return r

# Setting 0 engines for balloons
df['Number.of.Engines'][df['Number.of.Engines'].isnull() & (df['Make']
.str.contains('balloon', case=False))] = 0.0
# Correcting number of engines
num_engines = df.apply(lambda x: fix_number_of_engines(x['Number.of.En
gines'], x['Make']), axis=1)
df = df.assign(NumberofEngines = num_engines, index=df.index)
# Still some null after number of engines correction
df['NumberofEngines'].fillna(1, inplace=True)
```

```
/opt/conda/lib/python3.5/site-packages/ipykernel/__main__.py:11: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

Engine types

Taking engine types from the rest of the data

In [13]:

```
# Function that fixes the engine types
def fix_engine_type(et, model):
    if type(et) == str:
        return et
    else:
        # Setting engine type at the mode of engines for the model
        e = (df['Engine.Type'][df['Model']==model].mode())
        return  e[0] if e.count() > 0 else 'Unknown'
# Fix for Engine.Type field
en_type = df.apply(lambda x: fix_engine_type(x['Engine.Type'], x['Mode
l']), axis=1)
df = df.assign(EngineType = en_type, index=df.index)
```

```
df = df.assign(EngineType = en_type, index=df.index)
```

Aircraft Category

Taking Aircraft Categories from the rest of the data

In [14]:
```
# Function that fixes the Aircraft.Category
def fix_aircraft_category(cat, model):
    if type(cat) == str:
        return cat
    else:
        # Setting aircraft category at the mode of caterogories for the
model
        e = (df['Aircraft.Category'][df['Model']==model].mode())
        return  e[0] if e.count() > 0 else 'Unknown'
# Fix for Aircraft.Category field
aircraft_cat = df.apply(lambda x: fix_aircraft_category(x['Aircraft.Ca
tegory'], x['Model']), axis=1)
df = df.assign(AircraftCategory = aircraft_cat, index=df.index)
```

Country

It seems that null countries are all outside U.S.

In [15]:
```
# null countries are outside US
df[df['Country'].isnull()]
df['Country'].fillna('Foreign', inplace=True)
```

Injuries

I add a colunm that represents the total number of injuries in the accidents.

In [16]:
```
df['Injuries'] = df['Total.Fatal.Injuries'] + df['Total.Serious.Injuri
es'] + df['Total.Minor.Injuries']
```

Checking if all nulls have been fixed

In [17]:
```
#category_values(df, ['AircraftCategory', 'Country', 'EngineType', 'Numb
erofEngines', 'AmateurBuilt'])
#df['EngineType'].sample(100)

#df.groupby(by=['Location']).count()
df.isnull().sum()
```

Out[17]:
```
Event.Id                     0
Investigation.Type           0
Accident.Number              0
Event.Date                   0
Location                     0
Country                      0
Latitude                 53537
```

```
Longitude               53546
Airport.Code                0
Airport.Name                0
Injury.Severity             0
Aircraft.Damage             0
Aircraft.Category           0
Registration.Number      3084
Make                        0
Model                       0
Amateur.Built             572
Number.of.Engines        3985
Engine.Type              3373
FAR.Description             0
Schedule                    0
Purpose.of.Flight           0
Air.Carrier                 0
Total.Fatal.Injuries    23309
Total.Serious.Injuries  25550
Total.Minor.Injuries    24458
Total.Uninjured         12342
Weather.Condition           0
Broad.Phase.of.Flight       0
Report.Status               0
Publication.Date        13473
Year                        0
Month                       0
Day                         0
AmateurBuilt                0
index                       0
NumberofEngines             0
EngineType                  0
AircraftCategory            0
Injuries                29555
dtype: int64
```

Dropping columns that I will not use

There are some columns that I think are not so useful and others that have been replaced by "sanitized" ones.

In [18]:

```
df = df.drop(['Number.of.Engines', 'Aircraft.Category', 'Engine.Type',
'Amateur.Built', 'index'], axis='columns')
df = df.drop(['Publication.Date'], axis='columns')
```

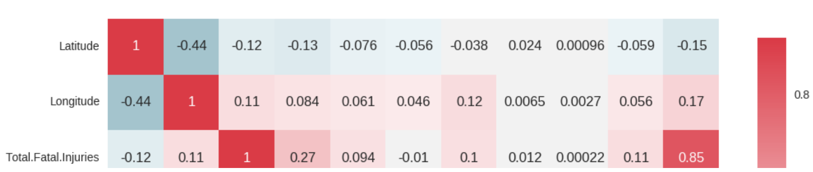
Now some visualization

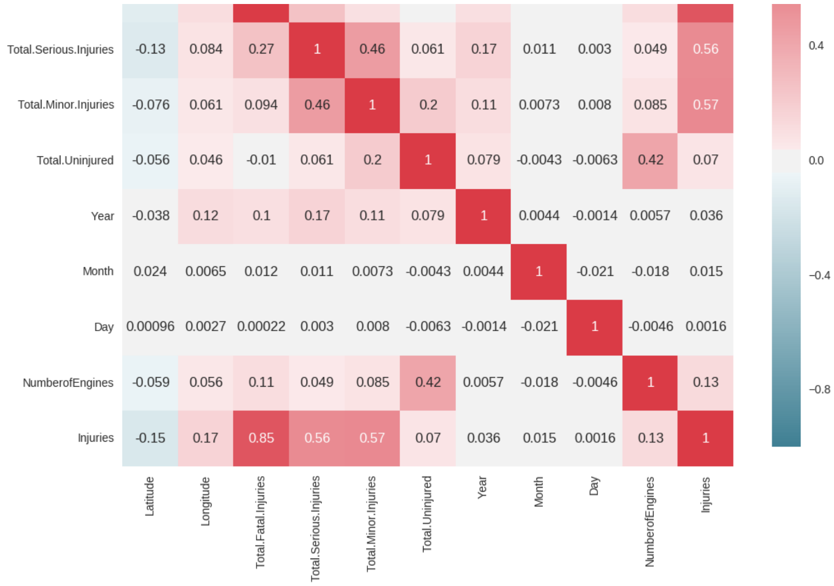
A better way to understand what's inside the data is to put some features in charts.

TODO: comment

In [19]:

```
plot_correlation_map(df)
```



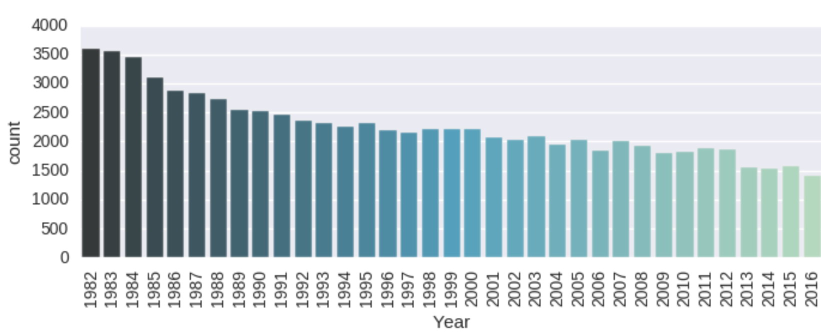


An observation: the number of uninjuried seems to be very related to the number of engines. Could it mean that a second engine helps in some kind of accident?

Time series charts

Let's see on the timeline some events.

In [20]:
```
# For the time series charts I start sorting data
df = df.sort_values(by=['Year', 'Month', 'Day'], ascending=True)

years = np.arange(1982, 2017)

sns.set(style="darkgrid")

plt.subplot(211)

g = sns.countplot(x="Year", data=df, palette="GnBu_d", order=years)
g.set_xticklabels(labels=years)
a = plt.setp(g.get_xticklabels(), rotation=90)
```



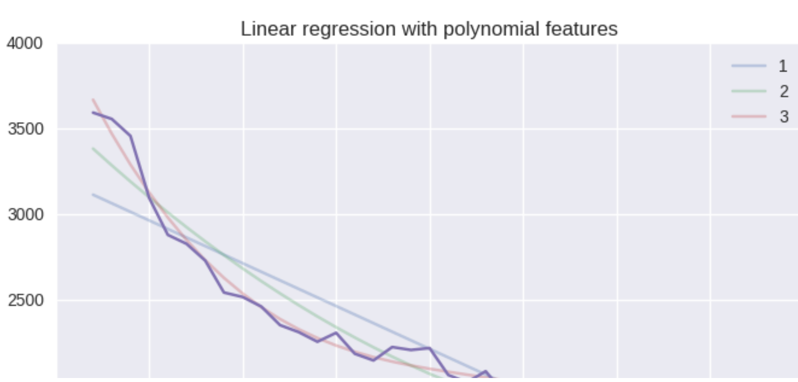
Linear regression on number of incidents

Given the histogram before, it should be easy to make a linear regression to predict next years' incidents.

In [21]:

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score


events_per_year = df.groupby(by='Year').count()['Event.Id']
events_per_year.drop(2017, axis=0, inplace=True)

X = [ [y] for y in events_per_year.index.values]
y = [ [e] for e in events_per_year.as_matrix()]


degrees = [1,2,3]
lr_pred_X = [[y] for y in range(1982, 2020)]
for i in range(len(degrees)):
    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X, y)

    # Evaluate the models using crossvalidation
    scores = cross_val_score(pipeline, X, y,
                             scoring="neg_mean_squared_error", cv=10)
    lr_pred=pipeline.predict(lr_pred_X)
    plt.plot(lr_pred_X, lr_pred, alpha=.3)

    print("Score for degree %d: %.3f - prediction for 2017 is %d" % (i
, pipeline.score(X, y), lr_pred[35]))

plt.plot(X, y)
plt.title("Linear regression with polynomial features")
plt.legend(labels=degrees)

plt.show()
```

```
Score for degree 0: 0.868 - prediction for 2017 is 1366
Score for degree 1: 0.926 - prediction for 2017 is 1685
Score for degree 2: 0.982 - prediction for 2017 is 1283
```



Avez-vous trouvé ce noyau utile?
Montrez votre appréciation avec un vote positif

sept

Les données

Source d'information

Base de données sur les accidents d'aviation et synopsis

AviationDataEnd2016UP.csv 31 colonnes

Base de données sur les accidents d'aviation et synopsis

L'ensemble de données sur les accidents d'aviation du NTSB

Dernière mise à jour: il y a 3 ans (version 1 de 8)

À propos de ce jeu de données

Contenu

The NTSB aviation accident database contains information from 1962 and later about civil aviation accidents and selected incidents within the United States, its territories and possessions, and in international waters.

Acknowledgements

Generally, a preliminary report is available online within a few days of an accident. Factual information is added when available, and when the investigation is completed, the preliminary report is replaced with a final description of the accident and its probable cause. Full narrative descriptions may not be available for dates before 1993, cases under revision, or where NTSB did not have primary investigative responsibility.

Inspiration

Hope it will teach us how to improve the quality and safety of traveling by Airplane.

Comments (3)

Sort by

All Comments | Hotness

Click here to comment...

Walid Salah • Publié sur la dernière version • il y a 3 ans • Options • Répondre 0

Merci d'avoir partagé

Kheirallah Samaha • Publié sur la dernière version • il y a 3 ans • Options • Répondre 0

Bon travail, j'aime ça.
À votre santé.

Dan Kernel Author • Publié sur la dernière version • il y a 3 ans • Options • Répondre 0

Je vous remercie!