IT6773 Practical Data Analytics- Team Project

Airline Performance Prediction:

Predict which airline is best/worst based on delay/cancellation/divert/accident history.

By Anthony Darden, Karis Kim, Khushbu Desai (May 3, 2019)

Part 0. About the data (Khushbu)

Flight (DOTBTS) Data:

25 CSV files around 59 MB each were downloaded.
Total Rows = 13,958,535
Total Columns = 24
Total Size of the Data = 2.5 GB
Data from 2017 Jan to 2019 Jan

Processing:

1. Removed some of the columns that were irrelevant to our model.
2. The average of the columns was calculated and grouped by Year, Month, OP_Carrier_Airline_ID (airlines).

Crash (NTBS) Data:

Total Rows = 44
Total Columns = 21
Total Size of the Data = 26 KB

Processing:

1. The only processing done for this file is to match the airlines (Air Carrier) from Crash Data with the Flight Data (OP_CARRIER_AIRLINE_ID), and add a Column that is common for both datasets.
2. Removed some of the columns that were irrelevant to our model.

Part I-A. Get the data (Khushbu)

Append all DOTBTS csv files into one dataframe

Cleanse/Aggregate DOTBTS files to show one row per airline per year-month with averages for Delay, Cancel, Divert

In [7]:
```
import pandas as pd

#FlightData (DOTBTS) Data
FD = pd.read_csv('D:\\SK KSU\\2019 Spr\\PracticalDataAnal\\PDA GrpProj\\Shared_w_Team\\GB_YM_ReqCol-w-AVG-&-SUM.csv', encoding='ISO-8859-1', header=0, delimiter= ',')
FD.head(5)
```

Out[7]:

	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARR_
0	2017	1	Alaska Airlines Inc.: AS	14711	0.793966	0.20
1	2017	1	American Airlines Inc.: AA	73132	0.805426	0.19
2	2017	1	Delta Air Lines Inc.: DL	69813	0.818323	0.18
3	2017	1	ExpressJet Airlines Inc.: EV	35037	0.755840	0.24
4	2017	1	Frontier Airlines Inc.: F9	7760	0.709065	0.29

In [8]:
```
print('\n FlightData (DOTBTS) Data \n')
FD.info()
```

```
 FlightData (DOTBTS) Data

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 368 entries, 0 to 367
Data columns (total 16 columns):
YEAR                    368 non-null int64
MONTH                   368 non-null int64
OP_CARRIER_AIRLINE_ID   368 non-null object
#FLIGHTS_PER_MONTHS     368 non-null int64
AVG_ARR_ONTIME          368 non-null float64
AVG_ARR_LATE            368 non-null float64
AVG_DEP_ONTIME          368 non-null float64
AVG_DEP_LATE            368 non-null float64
AVG_NOT_CANCELLED       368 non-null float64
AVG_CANCELLED           368 non-null float64
AVG_NOT_DIVERTED        368 non-null float64
AVG_DIVERTED            368 non-null float64
SUM_CARRIER_DELAY       368 non-null int64
SUM_WEATHER_DELAY       368 non-null int64
SUM_NAS_DELAY           368 non-null int64
SUM_SECURITY_DELAY      368 non-null int64
dtypes: float64(8), int64(7), object(1)
memory usage: 46.1+ KB
```

In [9]:
```
#CrashData (NTBS) Data
CD = pd.read_csv('D:\\SK KSU\\2019 Spr\\PracticalDataAnal\\PDA GrpProj\\Shared_w_Team\\NTBS_GroupBy_Y-M-i.csv', encoding='ISO-8859-1', header=0, delimiter= ',')
CD.head(5)
```

Out[9]:

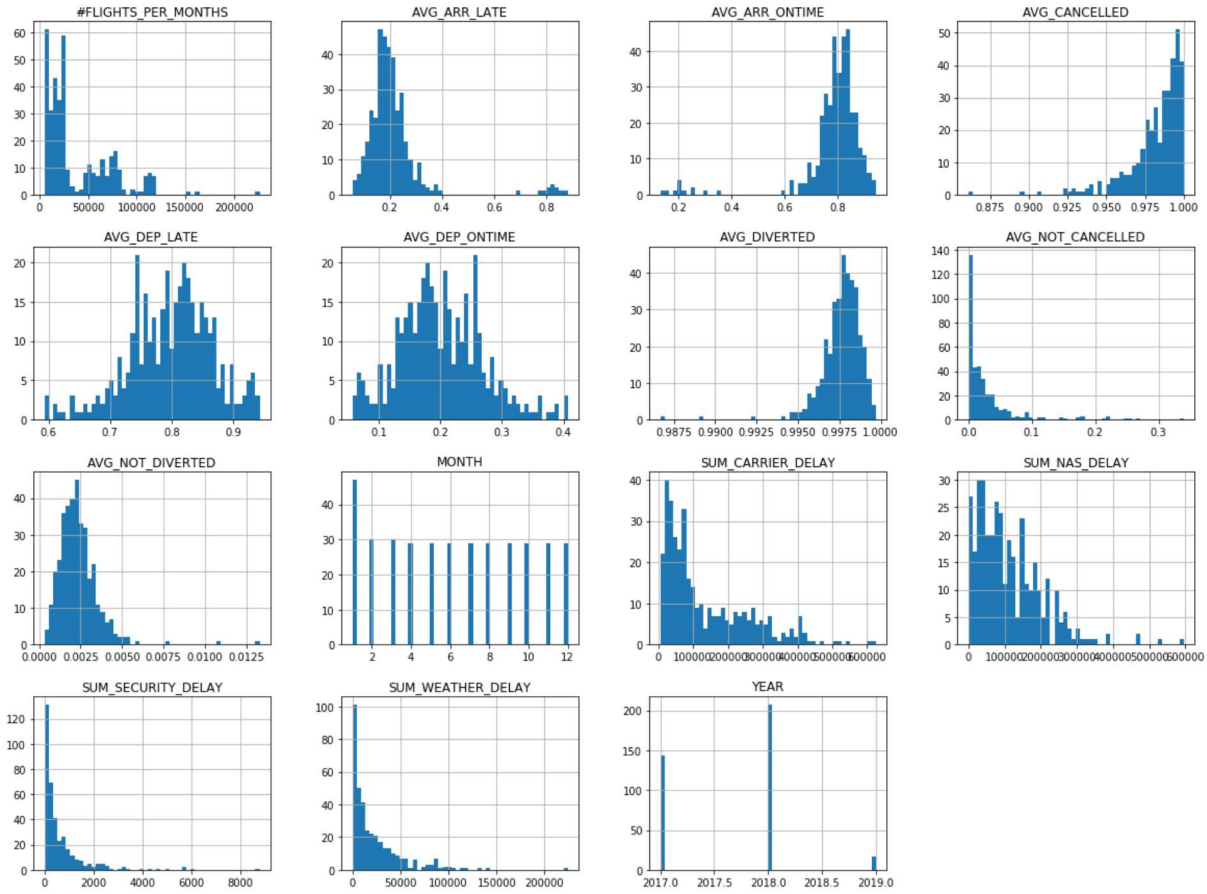
	YEAR	MONTH	AirCarrier	#FlightsPerMonth	InjurySeverity	AircraftDamage	TotalFatalInjuries	Tota
0	2017	1	Endeavor Air Inc.: 9E	1	1	0	0	
1	2017	2	Commutair	1	2	0	0	
2	2017	2	Southwest Airlines Co.: WN	1	2	0	0	
3	2017	3	Allegiant Air: G4	1	1	0	0	
4	2017	3	American Airlines Inc.: AA	1	2	1	0	

In [10]:
```
print('\n CRASH (NTBS) DATA \n')
print(CD.info())
```

```
 CRASH (NTBS) DATA

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 10 columns):
YEAR                    40 non-null int64
MONTH                   40 non-null int64
AirCarrier              40 non-null object
#FlightsPerMonth        40 non-null int64
InjurySeverity          40 non-null int64
AircraftDamage          40 non-null int64
TotalFatalInjuries      40 non-null int64
TotalSeriousInjuries    40 non-null int64
TotalMinorInjuries      40 non-null int64
TotalUninjured          40 non-null int64
dtypes: int64(9), object(1)
memory usage: 3.2+ KB
None
```

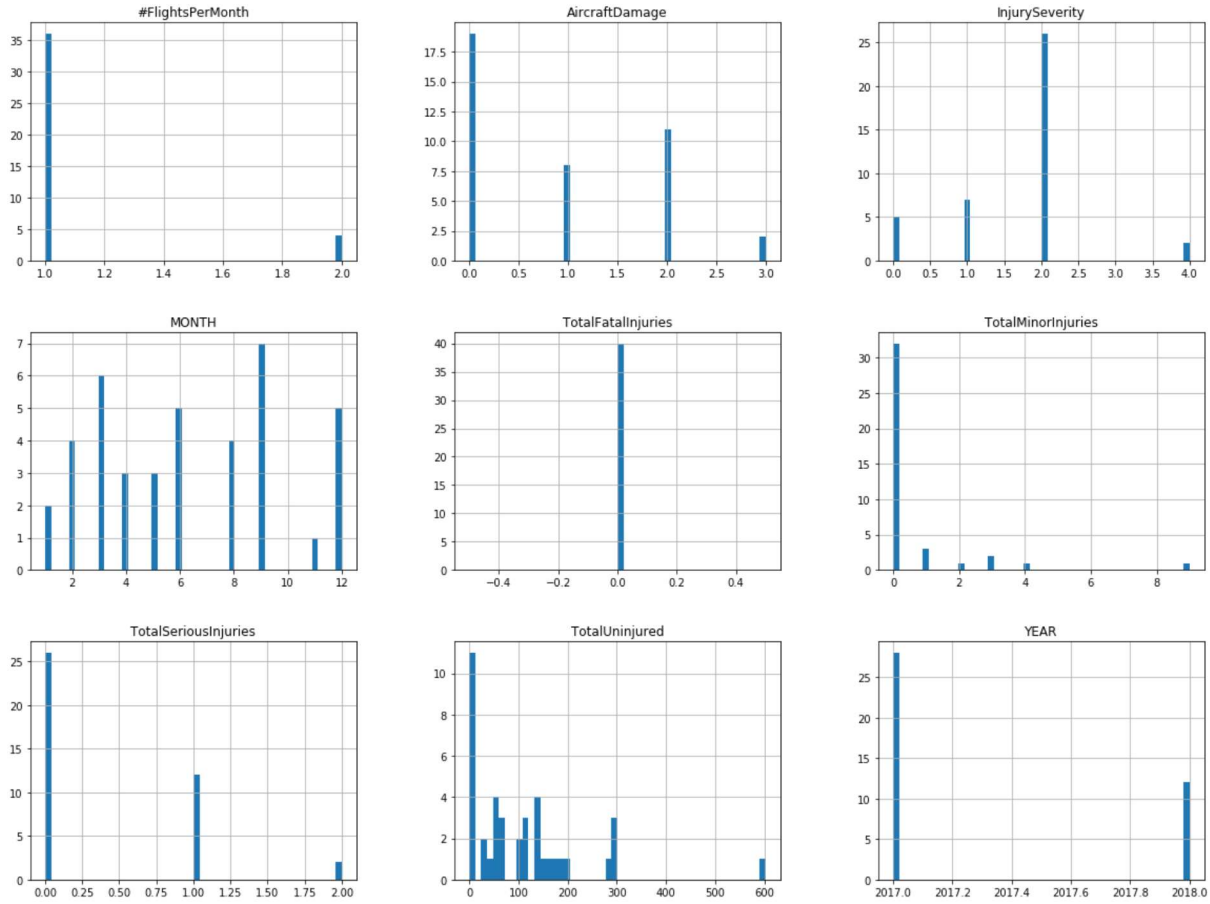
Part I-B. Explore the data (Khushbu)

In [11]:
```
#Visualize distribution of numerical attributes for flight data
%matplotlib inline
# This is executed on Terminal; is called Magic Function. It says how we want to sh
ow our plot.
# inline - means that we directly want to show in the
# matplotlib - is only available in Jupyter Notebook.
import matplotlib.pyplot as plt
FD.hist(bins=50, figsize=(20,15)) # Hist Plot is used because it helps to see the d
istribution view of data.
plt.show()

print('\n FlightData (DOTBTS) Data \n')
```



FlightData (DOTBTS) Data

In [12]:
```
#Visualize distribution of numerical attributes for crash data
%matplotlib inline
# This is executed on Terminal; is called Magic Function. It says how we want to sh
ow our plot.
# inline - means that we directly want to show in the
# matplotlib - is only available in Jupyter Notebook.
import matplotlib.pyplot as plt
CD.hist(bins=50, figsize=(20,15)) # Hist Plot is used because it helps to see the d
istribution view of data.
plt.show()

print('\n CRASH (NTBS) DATA \n')
```




```
CRASH (NTBS) DATA
```

Part II-A. Prepare for split (Karis)

Change air carrier column name in order to join two dataframes. Create target variable/predictor column "poor performance"

In [13]:
```
#First, change CD.Air Carrier column name to match FD column name for airline so th
at we can use it to join
CD.columns = ['YEAR', 'MONTH', 'OP_CARRIER_AIRLINE_ID', 'FlightsPerMonth', 'InjuryS
everity',
              'AircraftDamage', 'TotalFatalInjuries', 'TotalSeriousInjuries', 'Tota
lMinorInjuries',
              'TotalUninjured']
CD.head(5)
```

Out[13]:

	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	FlightsPerMonth	InjurySeverity	AircraftDamage	TotalFata
0	2017	1	Endeavor Air Inc.: 9E	1	1	0	
1	2017	2	Commutair	1	2	0	
2	2017	2	Southwest Airlines Co.: WN	1	2	0	
3	2017	3	Allegiant Air: G4	1	1	0	
4	2017	3	American Airlines Inc.: AA	1	2	1	

In [14]:
```
#Merge FD & CD by year, month, airline into df_merge
df_merge = pd.merge(FD, CD, how='left', on=['YEAR', 'MONTH', 'OP_CARRIER_AIRLINE_ID'])
df_merge.head(40)
```

Out[14]:

	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARR
0	2017	1	Alaska Airlines Inc.: AS	14711	0.793966	0.2
1	2017	1	American Airlines Inc.: AA	73132	0.805426	0.1
2	2017	1	Delta Air Lines Inc.: DL	69813	0.818323	0.1
3	2017	1	ExpressJet Airlines Inc.: EV	35037	0.755840	0.2
4	2017	1	Frontier Airlines Inc.: F9	7760	0.709065	0.2
5	2017	1	Hawaiian Airlines Inc.: HA	6276	0.860022	0.1
6	2017	1	JetBlue Airways: B6	24602	0.747121	0.2
7	2017	1	SkyWest Airlines Inc.: OO	50146	0.741926	0.2
8	2017	1	Southwest Airlines Co.: WN	107785	0.764247	0.2
9	2017	1	Spirit Air Lines: NK	12570	0.755822	0.2
10	2017	1	United Air Lines Inc.: UA	42403	0.790018	0.2
11	2017	1	Virgin America: VX	5782	0.663434	0.3
12	2017	10	Alaska Airlines Inc.: AS	15076	0.866934	0.1
13	2017	10	American Airlines Inc.: AA	75712	0.856402	0.1
14	2017	10	Delta Air Lines Inc.: DL	79476	0.895239	0.1
15	2017	10	ExpressJet Airlines Inc.: EV	23622	0.819262	0.1
16	2017	10	Frontier Airlines Inc.: F9	9270	0.826474	0.1
17	2017	10	Hawaiian Airlines Inc.: HA	6744	0.900000	0.1
18	2017	10	JetBlue Airways: B6	24285	0.815018	0.1
19	2017	10	SkyWest Airlines Inc.: OO	64740	0.831214	0.1
20	2017	10	Southwest Airlines Co.: WN	110262	0.851981	0.1
21	2017	10	Spirit Air Lines: NK	13080	0.877521	0.1
22	2017	10	United Air Lines Inc.: UA	51284	0.863625	0.1
23	2017	10	Virgin America: VX	6246	0.745804	0.2
24	2017	11	Alaska Airlines Inc.: AS	14638	0.841185	0.1
25	2017	11	American Airlines Inc.: AA	69677	0.890091	0.1
26	2017	11	Delta Air Lines Inc.: DL	74348	0.937513	0.0
27	2017	11	ExpressJet Airlines Inc.: EV	20595	0.866901	0.1
28	2017	11	Frontier Airlines Inc.: F9	9165	0.862194	0.1
29	2017	11	Hawaiian Airlines Inc.: HA	6500	0.897634	0.1
30	2017	11	JetBlue Airways: B6	23664	0.869202	0.1
31	2017	11	SkyWest Airlines Inc.: OO	59264	0.861926	0.1
32	2017	11	Southwest Airlines Co.: WN	109016	0.881121	0.1
33	2017	11	Spirit Air Lines: NK	12806	0.901773	0.0
34	2017	11	United Air Lines Inc.: UA	48354	0.886834	0.1
35	2017	11	Virgin America: VX	6135	0.789439	0.2
36	2017	12	Alaska Airlines Inc.: AS	15383	0.842773	0.1

	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARR
37	2017	12	American Airlines Inc.: AA	73744	0.824950	0.1
38	2017	12	Delta Air Lines Inc.: DL	72805	0.861909	0.1
39	2017	12	ExpressJet Airlines Inc.: EV	20759	0.779110	0.2

40 rows × 23 columns

In [15]:
```
#Description stats on combined dataset
print(df_merge.info())
df_merge.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 370 entries, 0 to 369
Data columns (total 23 columns):
YEAR                    370 non-null int64
MONTH                   370 non-null int64
OP_CARRIER_AIRLINE_ID   370 non-null object
#FLIGHTS_PER_MONTHS     370 non-null int64
AVG_ARR_ONTIME          370 non-null float64
AVG_ARR_LATE            370 non-null float64
AVG_DEP_ONTIME          370 non-null float64
AVG_DEP_LATE            370 non-null float64
AVG_NOT_CANCELLED       370 non-null float64
AVG_CANCELLED           370 non-null float64
AVG_NOT_DIVERTED        370 non-null float64
AVG_DIVERTED            370 non-null float64
SUM_CARRIER_DELAY       370 non-null int64
SUM_WEATHER_DELAY       370 non-null int64
SUM_NAS_DELAY           370 non-null int64
SUM_SECURITY_DELAY      370 non-null int64
FlightsPerMonth         29 non-null float64
InjurySeverity          29 non-null float64
AircraftDamage          29 non-null float64
TotalFatalInjuries      29 non-null float64
TotalSeriousInjuries    29 non-null float64
TotalMinorInjuries      29 non-null float64
TotalUninjured          29 non-null float64
dtypes: float64(15), int64(7), object(1)
memory usage: 69.4+ KB
None
```

Out[15]:

	YEAR	MONTH	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARR_LATE	AVG_D
count	370.000000	370.000000	370.000000	370.000000	370.000000	
mean	2017.651351	6.216216	38096.235135	0.786612	0.214724	
std	0.565552	3.569178	33815.684975	0.118868	0.125059	
min	2017.000000	1.000000	5140.000000	0.131844	0.056523	
25%	2017.000000	3.000000	14087.000000	0.764328	0.158960	
50%	2018.000000	6.000000	23963.000000	0.805777	0.194223	
75%	2018.000000	9.000000	62071.000000	0.841040	0.235672	
max	2019.000000	12.000000	226478.000000	0.943477	0.882832	

8 rows × 22 columns

In [16]:
```
#List of unique airlines in dataset
print(df_merge['OP_CARRIER_AIRLINE_ID'].unique())
print("Total number of unique airlines: ", len(df_merge['OP_CARRIER_AIRLINE_ID'].un
ique().tolist()))
```

```
['Alaska Airlines Inc.: AS' 'American Airlines Inc.: AA'
 'Delta Air Lines Inc.: DL' 'ExpressJet Airlines Inc.: EV'
 'Frontier Airlines Inc.: F9' 'Hawaiian Airlines Inc.: HA'
 'JetBlue Airways: B6' 'SkyWest Airlines Inc.: OO'
 'Southwest Airlines Co.: WN' 'Spirit Air Lines: NK'
 'United Air Lines Inc.: UA' 'Virgin America: VX' 'Allegiant Air: G4'
 'Endeavor Air Inc.: 9E' 'Envoy Air: MQ' 'Mesa Airlines Inc.: YV'
 'PSA Airlines Inc.: OH' 'Republic Airline: YX']
Total number of unique airlines:  18
```

Derive column "Poor Performance" to sum up columns w/metrics for poor performance.

This will be the predictor/label column by which airlines will be evaluated.

In [17]:

```
df_merge["poorPerform"] = df_merge[['InjurySeverity','AircraftDamage','TotalFatalInjuries','TotalSeriousInjuries','TotalMinorInjuries', 'AVG_ARR_LATE', 'AVG_DEP_LATE', 'AVG_CANCELLED', 'AVG_DIVERTED']].sum(axis=1, skipna=True)
#Leave NaN values in columns from CD, because it's different from examples/rows with 0 in CD- which had accident/incidents but the value was 0
#So,to sum poorPerform, use skipna=True to add only numerical values in that row

df_merge.head(40)
```

Out[17]:

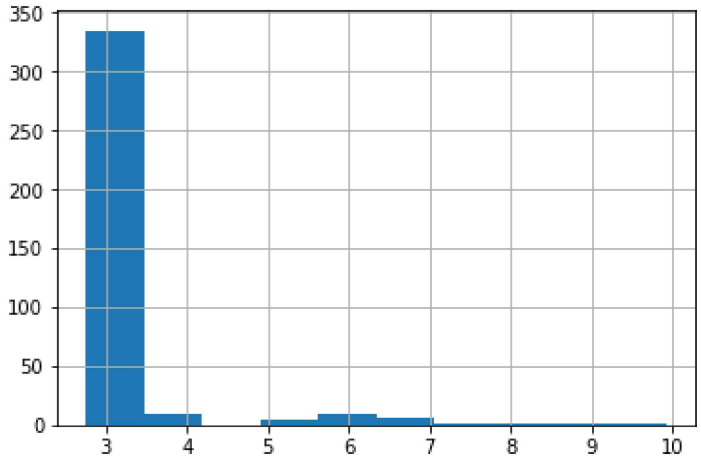
	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARR
0	2017	1	Alaska Airlines Inc.: AS	14711	0.793966	0.2
1	2017	1	American Airlines Inc.: AA	73132	0.805426	0.1
2	2017	1	Delta Air Lines Inc.: DL	69813	0.818323	0.1
3	2017	1	ExpressJet Airlines Inc.: EV	35037	0.755840	0.2
4	2017	1	Frontier Airlines Inc.: F9	7760	0.709065	0.2
5	2017	1	Hawaiian Airlines Inc.: HA	6276	0.860022	0.1
6	2017	1	JetBlue Airways: B6	24602	0.747121	0.2
7	2017	1	SkyWest Airlines Inc.: OO	50146	0.741926	0.2
8	2017	1	Southwest Airlines Co.: WN	107785	0.764247	0.2
9	2017	1	Spirit Air Lines: NK	12570	0.755822	0.2
10	2017	1	United Air Lines Inc.: UA	42403	0.790018	0.2
11	2017	1	Virgin America: VX	5782	0.663434	0.3
12	2017	10	Alaska Airlines Inc.: AS	15076	0.866934	0.1
13	2017	10	American Airlines Inc.: AA	75712	0.856402	0.1
14	2017	10	Delta Air Lines Inc.: DL	79476	0.895239	0.1
15	2017	10	ExpressJet Airlines Inc.: EV	23622	0.819262	0.1
16	2017	10	Frontier Airlines Inc.: F9	9270	0.826474	0.1
17	2017	10	Hawaiian Airlines Inc.: HA	6744	0.900000	0.1
18	2017	10	JetBlue Airways: B6	24285	0.815018	0.1
19	2017	10	SkyWest Airlines Inc.: OO	64740	0.831214	0.1
20	2017	10	Southwest Airlines Co.: WN	110262	0.851981	0.1
21	2017	10	Spirit Air Lines: NK	13080	0.877521	0.1
22	2017	10	United Air Lines Inc.: UA	51284	0.863625	0.1
23	2017	10	Virgin America: VX	6246	0.745804	0.2
24	2017	11	Alaska Airlines Inc.: AS	14638	0.841185	0.1
25	2017	11	American Airlines Inc.: AA	69677	0.890091	0.1
26	2017	11	Delta Air Lines Inc.: DL	74348	0.937513	0.0
27	2017	11	ExpressJet Airlines Inc.: EV	20595	0.866901	0.1
28	2017	11	Frontier Airlines Inc.: F9	9165	0.862194	0.1
29	2017	11	Hawaiian Airlines Inc.: HA	6500	0.897634	0.1
30	2017	11	JetBlue Airways: B6	23664	0.869202	0.1
31	2017	11	SkyWest Airlines Inc.: OO	59264	0.861926	0.1
32	2017	11	Southwest Airlines Co.: WN	109016	0.881121	0.1
33	2017	11	Spirit Air Lines: NK	12806	0.901773	0.0
34	2017	11	United Air Lines Inc.: UA	48354	0.886834	0.1
35	2017	11	Virgin America: VX	6135	0.789439	0.2
36	2017	12	Alaska Airlines Inc.: AS	15383	0.842773	0.1

	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARR
37	2017	12	American Airlines Inc.: AA	73744	0.824950	0.1
38	2017	12	Delta Air Lines Inc.: DL	72805	0.861909	0.1
39	2017	12	ExpressJet Airlines Inc.: EV	20759	0.779110	0.2

40 rows × 24 columns

Visualize the derived column "Poor Performance" score

In [18]:
```
#visualize distribution of numerical attributes
%matplotlib inline
# this is executed on Terminal and it is called Magic Function.It says how we want
 to show our Plot
# inline - means that we directly want to show in the
# matplotlin - is only available in Jupyter Notebook.
import matplotlib.pyplot as plt
#df_merge.hist(bins=50, figsize=(20,15)) # Hist Plot is used because it helps to se
e the distribution view of data.
df_merge["poorPerform"].hist()
plt.show()

print('\n Deprived Column ["PoorPerform"] \n')
```




```
Deprived Column ["PoorPerform"]
```

Part II-B. Split into training and testing (Karis)

In [19]:
```
import numpy as np
from sklearn.model_selection import train_test_split

#X is dataset, y is target/predictor
X = df_merge
y = df_merge.poorPerform
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.2, random_sta
te=42) #random_state to replicate split at iterations

print("Training set rows, columns: ", train_X.shape)
print("Testing set rows, columns: ",test_X.shape)
print("Training set labels top 10: ", train_y.head(10))
print("Testing set labels top 10: ", test_y.head(10))
```

```
Training set rows, columns:  (296, 24)
Testing set rows, columns:  (74, 24)
Training set labels top 10:  345    2.961405
192    2.958900
75     2.966425
84     3.007654
358    3.659322
16     2.977903
66     5.955380
283    2.938944
7      2.955287
113    2.978569
Name: poorPerform, dtype: float64
Testing set labels top 10:  327    4.997516
33     2.993706
15     2.982801
314    2.956289
57     2.973661
239    3.026622
76     3.014860
119    2.929900
305    2.997611
126    2.935215
Name: poorPerform, dtype: float64
```

In [20]:
```
print("Training set: ")
train_X.head()
```

```
Training set:
```

Out[20]:

	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARI
345	2019	1	Endeavor Air Inc.: 9E	20198	0.797538	0
192	2018	12	ExpressJet Airlines Inc.: EV	13882	0.738654	0
75	2017	5	ExpressJet Airlines Inc.: EV	61702	0.788943	0
84	2017	6	Alaska Airlines Inc.: AS	16378	0.835651	0
358	2017	3	Alaska Airlines Inc.: AS	15422	0.204386	0

5 rows × 24 columns

Set target attribute and save label separately

In [21]:
```
#Change target attribute to "poorPerform"
airline = train_X.drop("poorPerform", axis=1) #Drop labels for training set with in
put attributes only
airline.head()
```

Out[21]:

	YEAR	MONTH	OP_CARRIER_AIRLINE_ID	#FLIGHTS_PER_MONTHS	AVG_ARR_ONTIME	AVG_ARI
345	2019	1	Endeavor Air Inc.: 9E	20198	0.797538	0
192	2018	12	ExpressJet Airlines Inc.: EV	13882	0.738654	0
75	2017	5	ExpressJet Airlines Inc.: EV	61702	0.788943	0
84	2017	6	Alaska Airlines Inc.: AS	16378	0.835651	0
358	2017	3	Alaska Airlines Inc.: AS	15422	0.204386	0

5 rows × 23 columns

In [22]:
```
airline_labels = train_X["poorPerform"].copy() #Here's where labels are
print("Airline labels: ")
airline_labels.head()
```

```
Airline labels:
```

Out[22]:
```
345    2.961405
192    2.958900
75     2.966425
84     3.007654
358    3.659322
Name: poorPerform, dtype: float64
```

Part III. Preprocess the training data (Karis)

Machine learning models don't like missing values NaN, so replace with 0

In [23]:
```
from sklearn.pipeline import Pipeline #To combine preprocessing into one fit_transf
orm pipeline
from sklearn.impute import SimpleImputer #To deal with missing values

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy='constant', fill_value=0)),
    ])
```

In [24]:
```
try:
    from sklearn.compose import ColumnTransformer
except ImportError:
    from future_encoders import ColumnTransformer # Scikit-Learn < 0.20
```

In [25]:
```
from sklearn.preprocessing import OneHotEncoder

attribs = list(airline)
cat_attribs = ["OP_CARRIER_AIRLINE_ID"]
num_attribs = [x for x in attribs if x not in cat_attribs]

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attribs),
        ("cat", OneHotEncoder(), cat_attribs), #OneHotEncoder to transform cat valu
es to numerical without imposing ordinal bias
    ])

airline_prepared = full_pipeline.fit_transform(airline)
```

Part IV. Select and train a model (Anthony)

Try different models on training data itself- but I would assume there's a linear relationship, so I propose LinReg

In [26]:
```
from sklearn.metrics import mean_squared_error
```

In [27]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(airline_prepared, airline_labels)

airline_predictions = lin_reg.predict(airline_prepared)
lin_mse = mean_squared_error(airline_labels, airline_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

Out[27]:
```
4.350124359869019e-11
```

In [28]:
```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(random_state=42)
tree_reg.fit(airline_prepared, airline_labels)

airline_predictions = tree_reg.predict(airline_prepared)
tree_mse = mean_squared_error(airline_labels, airline_predictions)
tree_rmse = np.sqrt(tree_mse)
tree_rmse
```

Out[28]:
```
8.868833423263083e-05
```

In [29]:
```
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=10, random_state=42)
forest_reg.fit(airline_prepared, airline_labels)

airline_predictions = forest_reg.predict(airline_prepared)
forest_mse = mean_squared_error(airline_labels, airline_predictions)
forest_rmse = np.sqrt(forest_mse)
forest_rmse
```

Out[29]:
```
0.17154315322597757
```

In [30]:
```
from sklearn.svm import SVR
import numpy as np

svm_reg = SVR(kernel="linear")
svm_reg.fit(airline_prepared, airline_labels)
housing_predictions = svm_reg.predict(airline_prepared)
svm_mse = mean_squared_error(airline_labels, airline_predictions)
svm_rmse = np.sqrt(svm_mse)
svm_rmse
```

Out[30]:
```
0.17154315322597757
```

In [31]:
```
print(lin_rmse)  # Best Fitting Model
print(tree_rmse)
print(forest_rmse)
print(svm_rmse)
```

```
4.350124359869019e-11
8.868833423263083e-05
0.17154315322597757
0.17154315322597757
```

Try cross_val on different models

In [32]:
```
from sklearn.model_selection import cross_val_score
```

In [33]:
```
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard deviation:", scores.std())
```

In [34]:
```
lin_scores = cross_val_score(lin_reg, airline_prepared, airline_labels,
scoring="neg_mean_squared_error", cv=10)
lin_rmse_scores = np.sqrt(-lin_scores)
display_scores(lin_rmse_scores)
```

```
Scores: [3.89679354e-11 3.26572885e-11 3.53409832e-11 9.02636333e-11
 5.40595012e-11 2.13420148e-11 2.28521868e-11 1.87686538e-11
 4.57847990e-11 2.36154140e-11]
Mean: 3.8365240983754733e-11
Standard deviation: 2.0417937148946064e-11
```

In [35]:
```
tree_scores = cross_val_score(tree_reg, airline_prepared, airline_labels,
scoring="neg_mean_squared_error", cv=10)
tree_rmse_scores = np.sqrt(-tree_scores)
display_scores(tree_scores)
```

```
Scores: [-0.12442039 -0.03482233 -0.30274011 -0.03553065 -0.29583118 -0.09071795
 -0.01946976 -0.02911913 -0.17647154 -0.00166864]
Mean: -0.11107916691770585
Standard deviation: 0.1069294301429971
```

Note: we specify `n_estimators=10` to avoid a warning about the fact that the default value is going to change to 100 in Scikit-Learn 0.22.

In [36]:
```
forest_scores = cross_val_score(forest_reg, airline_prepared, airline_labels,
scoring="neg_mean_squared_error", cv=10)
forest_rmse_scores = np.sqrt(-forest_scores)
display_scores(forest_rmse_scores)
```

```
Scores: [0.2346786  0.07191257 0.59072041 0.0422937  0.57015514 0.26207853
 0.14031606 0.10405612 0.34507781 0.11825784]
Mean: 0.2479546781460303
Standard deviation: 0.18801902236782525
```

In [39]:
```
svm_scores = cross_val_score(svm_reg, airline_prepared, airline_labels,
scoring="neg_mean_squared_error", cv=10)
svm_rmse_scores = np.sqrt(-svm_scores)
display_scores(svm_rmse_scores)
```

```
Scores: [ 5150.97775276  9968.59823292 12900.20026712  5665.81334308
  4320.49590422  5315.61647737 12569.00267372  8546.54839347
  6796.09808069  7612.00149445]
Mean: 7884.535261981466
Standard deviation: 2912.704314821858
```

In [40]:
```
#Bird's eye view
print(display_scores(lin_rmse_scores)) #BEST
print(display_scores(tree_scores))
print(display_scores(forest_rmse_scores))
print(display_scores(svm_rmse_scores))
```

```
Scores: [3.89679354e-11 3.26572885e-11 3.53409832e-11 9.02636333e-11
 5.40595012e-11 2.13420148e-11 2.28521868e-11 1.87686538e-11
 4.57847990e-11 2.36154140e-11]
Mean: 3.8365240983754733e-11
Standard deviation: 2.0417937148946064e-11
None
Scores: [-0.12442039 -0.03482233 -0.30274011 -0.03553065 -0.29583118 -0.09071795
 -0.01946976 -0.02911913 -0.17647154 -0.00166864]
Mean: -0.11107916691770585
Standard deviation: 0.1069294301429971
None
Scores: [0.2346786  0.07191257 0.59072041 0.0422937  0.57015514 0.26207853
 0.14031606 0.10405612 0.34507781 0.11825784]
Mean: 0.2479546781460303
Standard deviation: 0.18801902236782525
None
Scores: [ 5150.97775276  9968.59823292 12900.20026712  5665.81334308
  4320.49590422  5315.61647737 12569.00267372  8546.54839347
  6796.09808069  7612.00149445]
Mean: 7884.535261981466
Standard deviation: 2912.704314821858
None
```

Part V. Fine tune the best model (Anthony)

The best hyperparameter combination found:

In [41]:
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {
        'n_jobs': randint(low=-1, high=1),
    }

lin_reg = LinearRegression(fit_intercept = True)
rnd_search = RandomizedSearchCV(lin_reg, param_distributions=param_distribs,
                                n_iter=10, cv=10, scoring='neg_mean_squared_error',
random_state = 42)
rnd_search.fit(airline_prepared, airline_labels)
```

Out[41]:
```
RandomizedSearchCV(cv=10, error_score='raise-deprecating',
          estimator=LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False),
          fit_params=None, iid='warn', n_iter=10, n_jobs=None,
          param_distributions={'n_jobs': <scipy.stats._distn_infrastructure.rv_fro
zen object at 0x00000131D9960EB8>},
          pre_dispatch='2*n_jobs', random_state=42, refit=True,
          return_train_score='warn', scoring='neg_mean_squared_error',
          verbose=0)
```

In [42]:
```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
4.3616368949625416e-11 {'n_jobs': -1}
4.3616368949625416e-11 {'n_jobs': 0}
4.3616368949625416e-11 {'n_jobs': -1}
4.3616368949625416e-11 {'n_jobs': -1}
4.3616368949625416e-11 {'n_jobs': -1}
4.3616368949625416e-11 {'n_jobs': 0}
4.3616368949625416e-11 {'n_jobs': -1}
4.3616368949625416e-11 {'n_jobs': -1}
4.3616368949625416e-11 {'n_jobs': -1}
4.3616368949625416e-11 {'n_jobs': 0}
```

Part VI. Apply test to data (Khushbu)

In [43]:

```
import numpy as np
final_model = rnd_search.best_estimator_ #final_model to be used to call best metho
d

t_airline = test_X.drop("poorPerform", axis=1) #Drop labels for testing set with in
put attributes only
t_airline.head(10)

t_airline_labels = test_X["poorPerform"].copy() #Here's where labels are
print("Test Airline labels: ")
print(t_airline_labels)

t_airline_prepared = full_pipeline.transform(t_airline) #Transform testing data lik
e training data

final_predictions = final_model.predict(t_airline_prepared) #final_predictions= tra
nsformed testing set with best model applied

final_mse = mean_squared_error(t_airline_labels, final_predictions)
final_rmse = np.sqrt(final_mse)


print('Final Model RMSE = ',final_rmse)
print('t_airline_prepared',t_airline_prepared.shape)
print('t_airline_labels',t_airline_labels.shape)
print('Final_Predictions',final_predictions.shape)
```

Test Airline labels:
327   4.997516
33   2.993706
15   2.982801
314   2.956289
57   2.973661
239   3.026622
76   3.014860
119   2.929900
305   2.997611
126   2.935215
233   2.868116
39   2.985244
153   2.970232
55   2.968827
155   2.998935
278   2.940871
0   3.002104
231   2.961508
334   2.949511
101   2.976857
9   2.949840
180   2.972406
72   3.030690
237   2.958604
255   3.011522
137   2.932419
225   2.845065
194   3.029920
193   2.986105
366   3.606066
         ...
109   3.002961
124   3.947684
90   2.947044
56   2.973409
5   3.035323
45   6.994726
351   2.965677
145   2.863592
286   2.997546
218   2.985986
250   2.965150
342   2.966760
93   4.958988
3   2.938523
349   3.021950
77   3.019437
132   5.945888
310   2.982719
78   5.931734
46   2.999600
220   3.010818
73   3.010513
229   2.999237
326   8.985045
208   2.928951
82   2.999844
94   2.936975
195   2.993558

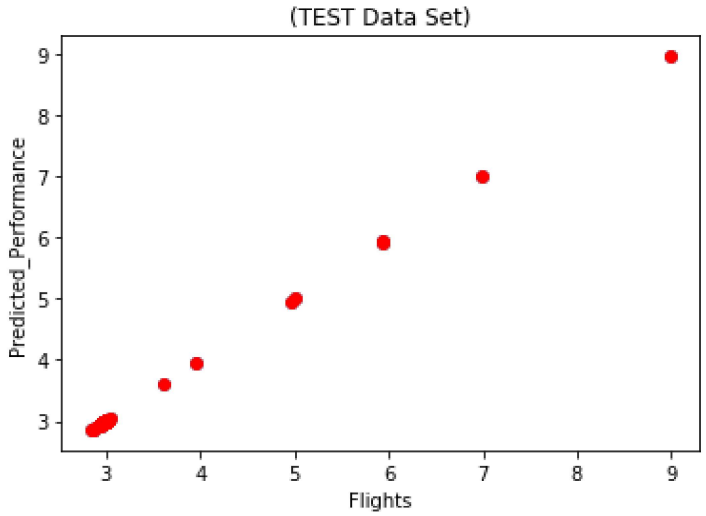
```
311    2.886362
292    2.976506
Name: poorPerform, Length: 74, dtype: float64
Final Model RMSE =  3.958696864264348e-11
t_airline_prepared (74, 40)
t_airline_labels (74,)
Final_Predictions (74,)
```

Part VII. Visualize prediction results (Karis & Anthony)

In [44]:
```
import numpy as np
import matplotlib.pyplot as plt  # To visualize
import pandas as pd
from sklearn.linear_model import LinearRegression
```

In [45]:
```
X = final_predictions #Put resulting testing set that had model applied in X
Y = t_airline_labels #Put testing labels in Y
final_model.fit(t_airline_prepared, Y)  # perform linear regression
Y_pred = final_model.predict(t_airline_prepared)  # make predictions
```

In [46]:
```
%matplotlib inline
plt.scatter(X, Y)
plt.plot(X, Y_pred, 'o', color='red')
plt.title('(TEST Data Set)')
plt.xlabel('Flights') #Name x axis of plot
plt.ylabel('Predicted_Performance') #Name y axis of plot
plt.show()
```



In [47]:
```
#Results of Prediction with Airline Name

Result = pd.DataFrame(Y_pred)

#Merging prediction with original data
df_merge['Predicted'] = Result

#DataFrame of Predicted Values and Airline Carriers
Final_Results = df_merge[["OP_CARRIER_AIRLINE_ID", "Predicted"]]

#There were more Airline Carriers in the df_merge data. This generates Nan in the p
redicted column.
#Create a dataframe that only displays data for columns that are not null.
Final_Results[Final_Results["Predicted"].notnull()]

test = Final_Results[Final_Results["Predicted"].notnull()].sort_values(['Predicted'
], ascending=[False],)
test.head()

print("Airlines predicted to perform the worst are: ")
```

```
Airlines predicted to perform the worst are:
```

Part VIII. Save the model (Khusbu)

In [49]:
```
from sklearn.externals import joblib
filename = 'final_model.pkl'
joblib.dump(final_model, 'final_model.pkl')

my_model_loaded = joblib.load("final_model.pkl")
```