CS 109A/STAT 121A/AC 209A/CSCI E-109A:

Midterm - 2017

Harvard University
Fall 2017
Instructors: Pavlos Protopapas, Kevin Rader, Rahul Dave, Margo Levine

---

INSTRUCTIONS

- You must submit the Midterm on your own. **No group submissions are allowed**. You may use any print or online resources but **you may not work or consult with others**.
- Restart the kernel and run the whole notebook again before you submit.
- Please submit both a notebook and a pdf.

In [2]:

```
import pandas as pd
import sys
import numpy as np
import statsmodels.api as sm
from statsmodels.tools import add_constant
from statsmodels.regression.linear_model import RegressionResults
import scipy as sp
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import LogisticRegression
import sklearn.metrics as metrics
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import LinearSVC
from sklearn.preprocessing import PolynomialFeatures
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.linear_model import LinearRegression

import seaborn as sns
import sklearn as sk
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
sns.set(style="ticks")
%matplotlib inline


import seaborn as sns
pd.set_option('display.width', 1500)
pd.set_option('display.max_columns', 600)
```

Flight Delays

The U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics tracks the on-time performance of domestic flights operated by large air carriers. Summary information on the number of on-time, delayed, canceled, and diverted flights are published in DOT's monthly Air Travel Consumer Report and in this dataset of 2015 flight delays and cancellations.

Data

Each entry of the flights.csv file corresponds to a flight. More than 5,800,000 flights were recorded in 2015. These flights are described according to 31 variables. Further details of these variables can be found here, if you are interested (not needed to answer these questions).

Name	Type	DESCRIPTION
DATE	object	The date in python datetime format
MONTH	int64	The month of the year(1-12)
DAY	int64	The day of the month
DAY_OF_WEEK	int64	The day of the week(1-7, MON-SUN)
AIRLINE	object	An identifier for the airline
FLIGHT_NUMBER	int64	The flight number
TAIL_NUMBER	object	The tail number (aircraft) corresponding to this flight
ORIGIN_AIRPORT	object	The code for origin airport
DESTINATION_AIRPORT	object	The code for destination airport
SCHED_DEP	object	The departure time in python datetime.time format
SCHED_ARR	object	The arrival time in python datetime.time format
DEPARTURE_DELAY	float64	The delay incurred at the origin (mins)
ARRIVAL_DELAY	float64	The delay when the flight reached the (mins) destination
DISTANCE	int64	Distance in miles between origin and destination
SCHEDULED_TIME	float64	Scheduled time of flight (minutes)
ELAPSED_TIME	float64	Actual time of flight (minutes)
AIR_SYSTEM_DELAY	float64	What part of the delay was NASD?(mins)
SECURITY_DELAY	float64	What part of the delay was due to security problems? (mins)
AIRLINE_DELAY	float64	What part of the delay is due to the airline? (mins)
LATE_AIRCRAFT_DELAY	float64	What part of the delay is due to previous flight(s) being late(mins)
WEATHER_DELAY	float64	Delay due to extreme weather events(min)

You can read more about the various weather delays [here](https://www.rita.dot.gov/bts/help/aviation/html/understanding.html) if you are so inclined.

Data/Caveats

The data file, flights.csv, is found here (note, it is about 70MB).

This data is already preprocessed, reduced, partially cleaned and therefore not identical to the original dataset.

In [33]:
```
# read in data
flights = pd.read_csv('cs109a_midterm.csv')
flights_2 = flights
flights = pd.get_dummies(flights, prefix = ['MONTH', 'DAY', 'DAY_OF_WEEK', 'ORIGIN_
```

In [ ]:

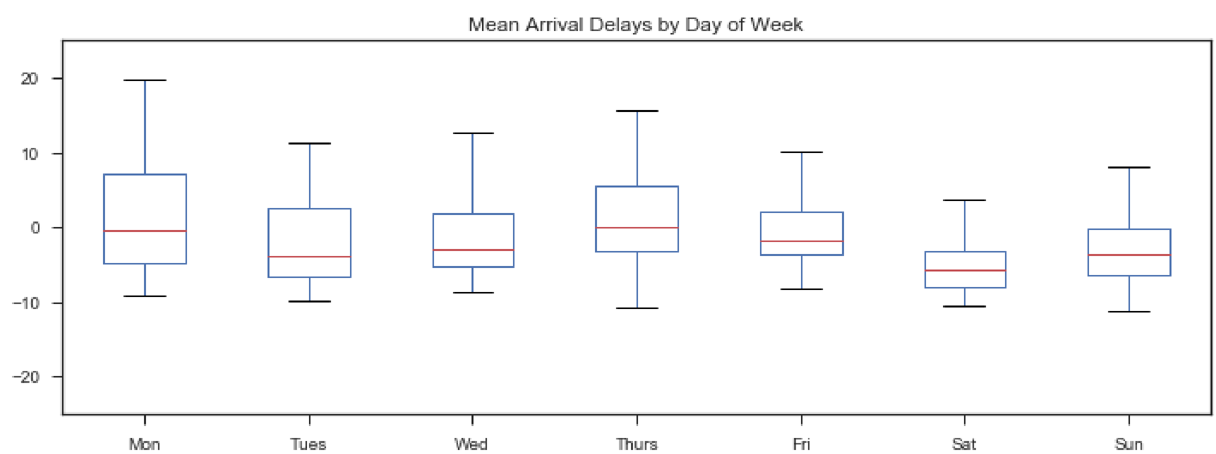
Problem Description

In [34]:
```
# visualization, EDA I focused a good bit on time, since I believe it might hold th
fig = plt.figure()

labels = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat','Sun']
ax = flights_2.pivot_table(index='DATE', columns='DAY_OF_WEEK')['ARRIVAL_DELAY'].p
ax.set_title("Mean Arrival Delays by Day of Week")
ax.set_ylabel="Delay in Minutes"
ax.set_xlabel="Day of Week"
ax.set_xticklabels(labels)
ax.set_ylim(-25,25)
```

Out[34]: (-25, 25)

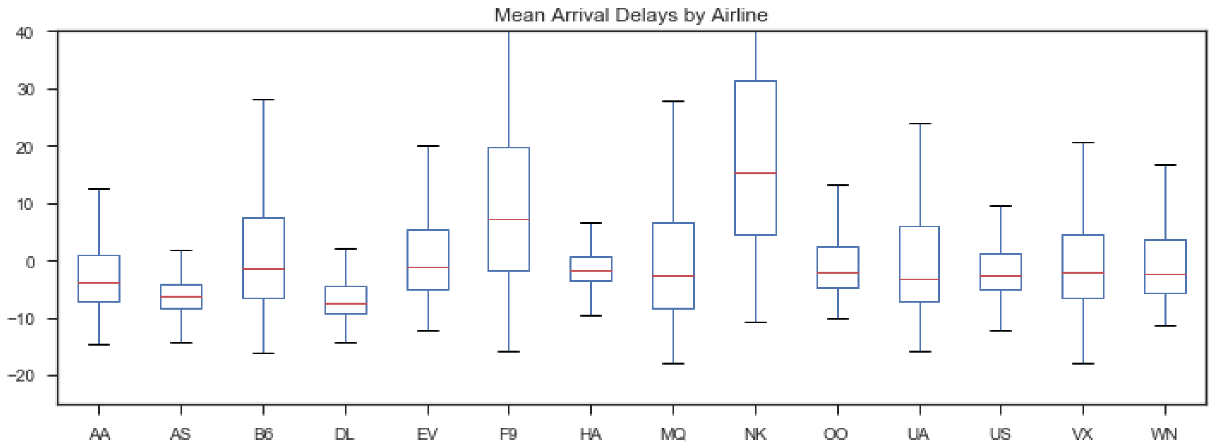
<matplotlib.figure.Figure at 0x1a48609b908>



In [35]:
```
fig = plt.figure()
ax1 = flights_2.pivot_table(index='DATE', columns='AIRLINE', values='ARRIVAL_DELAY
ax1.set_title("Mean Arrival Delays by Airline")
ax1.set_ylabel="Delay in Minutes"
ax1.set_xlabel="Day of Week"
ax1.set_ylim(-25,40)
```

Out[35]: (-25, 40)

<matplotlib.figure.Figure at 0x1a482d9ef98>

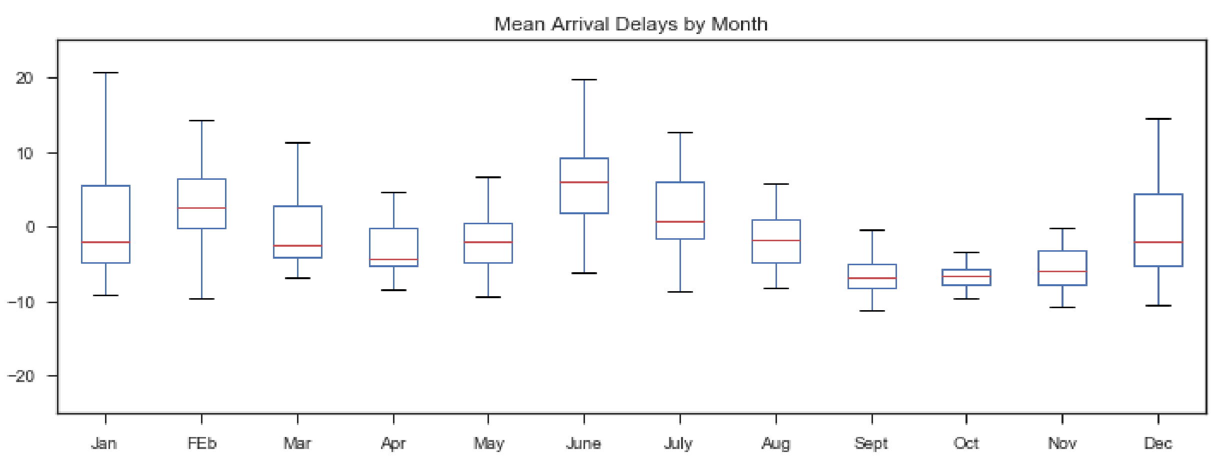


In [36]:
```
fig = plt.figure()

labels = ['Jan', 'FEb', 'Mar', 'Apr', 'May', 'June','July','Aug','Sept','Oct','Nov
ax = flights_2.pivot_table(index='DATE', columns='MONTH')['ARRIVAL_DELAY'].plot(ki
ax.set_title("Mean Arrival Delays by Month")
ax.set_ylabel="Delay in Minutes"
ax.set_xlabel="Month"
ax.set_xticklabels(labels)
ax.set_ylim(-25,25)
```

Out[36]: (-25, 25)

<matplotlib.figure.Figure at 0x1a4c1d8a128>

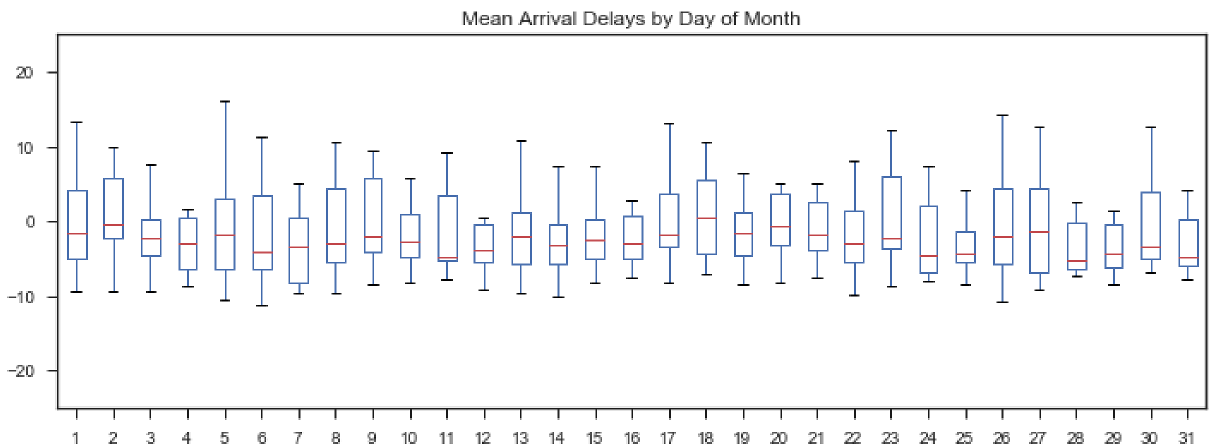


In [37]:
```
fig = plt.figure()

labels = ['1', '2', '3', '4', '5', '6','7','8','9','10','11', '12','13','14','15',
ax = flights_2.pivot_table(index='DATE', columns='DAY')['ARRIVAL_DELAY'].plot(kind
ax.set_title("Mean Arrival Delays by Day of Month")
ax.set_ylabel="Delay in Minutes"
ax.set_xlabel="Day"
ax.set_xticklabels(labels)
ax.set_ylim(-25,25)
```

Out[37]:
```
(-25, 25)
```

<matplotlib.figure.Figure at 0x1a4c1d057b8>



In [38]:
```
def late (row):
    if row['ARRIVAL_DELAY'] >= 15 :
        return 1
    if row['ARRIVAL_DELAY'] < 15 :
        return 0
    else :
        return 0

# add column
flights['DELAY_OR_NOT'] =  flights.apply(lambda row: late(row), axis=1)

# sample our data and subtract sampled data from full dataset
flights_sample = flights.sample(n=3000, random_state=4)
flights = flights.drop(flights_sample.index)
flights_sample = flights_sample.reset_index(drop=True)
```

In [39]:
```
flights_sample_2 = flights_sample
```

In [40]:
```
flights.describe()
```

Out[40]:

	FLIGHT_NUMBER	DEPARTURE_DELAY	ARRIVAL_DELAY	DISTANCE	SCHEDULED_TIME
count	801941.000000	801941.000000	801941.000000	801941.000000	801941.000000
mean	2158.241929	4.412873	-1.363375	815.308444	140.710637
std	1742.003060	27.265262	29.613119	599.360593	74.202443
min	1.000000	-56.000000	-79.000000	31.000000	18.000000
25%	736.000000	-5.000000	-14.000000	373.000000	85.000000
50%	1690.000000	-2.000000	-7.000000	645.000000	122.000000
75%	3172.000000	3.000000	3.000000	1050.000000	172.000000
max	7438.000000	1458.000000	1456.000000	4983.000000	718.000000

8 rows × 1312 columns

In [ ]:
```
```

In [41]:
```
# turn time data into integers 1-24
#def time(row):
#    return int(row['SCHED_DEP'][:2])
#
#flights_sample['SCHED_DEP'] =  flights_sample.apply(lambda row: time(row), axis=1,
#flights['SCHED_DEP'] =  flights.apply(lambda row: time(row), axis=1)
```

In [42]:
```
#def time_two(row):
#    return int(row['SCHED_ARR'][:2])
#
#flights_sample['SCHED_ARR'] =  flights_sample.apply(lambda row: time_two(row), ax
#flights['SCHED_ARR'] =  flights.apply(lambda row: time_two(row), axis=1)
```

In [43]:
```
flights_sample_2 = flights_sample
flights_2 = flights
```

In [44]:
```
# drop unecessary columns and seperate X and y
#y = ((np.array(flights_sample['DELAY_OR_NOT'])))
y = flights_sample['DELAY_OR_NOT'].values
flights_sample = flights_sample.drop(['ARRIVAL_DELAY','DATE','AIRLINE','SCHED_DEP'

X = flights_sample.loc[:].values

# split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_s

# drop unecessary columns and seperate X and y on full dataset
#y = ((np.array(flights_sample['DELAY_OR_NOT'])))
y_full = flights['DELAY_OR_NOT'].values
flights = flights.drop(['ARRIVAL_DELAY','DATE','AIRLINE','SCHED_DEP','SCHED_ARR','

X_full = flights.loc[:].values

# split into train and test
#X_train_full, X_test_full, y_train_full, y_test_full = train_test_split(X_full, y_
```

In [45]:
```
# check shape of data
y_train = y_train.reshape(len(y_train), 1)
y_test = y_test.reshape(len(y_test), 1)
print(np.shape(X_train), np.shape(y_train), np.shape(X_test), np.shape(y_test))
```

```
(2010, 1305) (2010, 1) (990, 1305) (990, 1)
```

In [46]:
```
# check shape of data of full dataset

y_full = y_full.reshape(len(y_full), 1)
print(np.shape(X_full), np.shape(y_full))
```

```
(801941, 1305) (801941, 1)
```

In [47]:
```
#check missingness
np.any(np.isnan(flights_sample))
```

Out[47]: False

In [48]:
```
#check missingness
np.all(np.isfinite(flights_sample))
```

Out[48]: True

In [20]:
```
#missingness summary
missing = flights.isnull().sum(axis=0).reset_index()
missing.columns = ['variable', 'missing values']
missing['percent_missing']=100-((flights.shape[0]-missing['missing values'])/fligh
missing.sort_values('percent_missing').reset_index(drop = True)
```

Out[20]:

	variable	missing values	percent_missing
0	DEPARTURE_DELAY	0	0.0
1	DESTINATION_AIRPORT_13367	0	0.0
2	DESTINATION_AIRPORT_13360	0	0.0
3	DESTINATION_AIRPORT_13344	0	0.0
4	DESTINATION_AIRPORT_13342	0	0.0
5	DESTINATION_AIRPORT_13303	0	0.0
6	DESTINATION_AIRPORT_13296	0	0.0
7	DESTINATION_AIRPORT_13290	0	0.0
8	DESTINATION_AIRPORT_13377	0	0.0
9	DESTINATION_AIRPORT_13277	0	0.0
10	DESTINATION_AIRPORT_13256	0	0.0
11	DESTINATION_AIRPORT_13244	0	0.0
12	DESTINATION_AIRPORT_13241	0	0.0
13	DESTINATION_AIRPORT_13232	0	0.0
14	DESTINATION_AIRPORT_13230	0	0.0
15	DESTINATION_AIRPORT_13204	0	0.0
16	DESTINATION_AIRPORT_13198	0	0.0
17	DESTINATION_AIRPORT_13264	0	0.0
18	DESTINATION_AIRPORT_13184	0	0.0
19	DESTINATION_AIRPORT_13422	0	0.0
20	DESTINATION_AIRPORT_13459	0	0.0
21	DESTINATION_AIRPORT_13931	0	0.0
22	DESTINATION_AIRPORT_13930	0	0.0
23	DESTINATION_AIRPORT_13891	0	0.0
24	DESTINATION_AIRPORT_13873	0	0.0
25	DESTINATION_AIRPORT_13871	0	0.0
26	DESTINATION_AIRPORT_13851	0	0.0
27	DESTINATION_AIRPORT_13830	0	0.0
28	DESTINATION_AIRPORT_13433	0	0.0
29	DESTINATION_AIRPORT_13796	0	0.0
...	...	...	...
1275	ORIGIN_AIRPORT_CHO	0	0.0

	variable	missing values	percent_missing
1276	ORIGIN_AIRPORT_CHA	0	0.0
1277	ORIGIN_AIRPORT_CEC	0	0.0
1278	ORIGIN_AIRPORT_CDV	0	0.0
1279	ORIGIN_AIRPORT_CDC	0	0.0
1280	ORIGIN_AIRPORT_CAK	0	0.0
1281	ORIGIN_AIRPORT_CAE	0	0.0
1282	ORIGIN_AIRPORT_CHS	0	0.0
1283	ORIGIN_AIRPORT_CNY	0	0.0
1284	ORIGIN_AIRPORT_COD	0	0.0
1285	ORIGIN_AIRPORT_COS	0	0.0
1286	ORIGIN_AIRPORT_DLG	0	0.0
1287	ORIGIN_AIRPORT_DIK	0	0.0
1288	ORIGIN_AIRPORT_DHN	0	0.0
1289	ORIGIN_AIRPORT_DFW	0	0.0
1290	ORIGIN_AIRPORT_DEN	0	0.0
1291	ORIGIN_AIRPORT_DCA	0	0.0
1292	ORIGIN_AIRPORT_DBQ	0	0.0
1293	ORIGIN_AIRPORT_DAY	0	0.0
1294	ORIGIN_AIRPORT_DAL	0	0.0
1295	ORIGIN_AIRPORT_DAB	0	0.0
1296	ORIGIN_AIRPORT_CWA	0	0.0
1297	ORIGIN_AIRPORT_CVG	0	0.0
1298	ORIGIN_AIRPORT_CSG	0	0.0
1299	ORIGIN_AIRPORT_CRW	0	0.0
1300	ORIGIN_AIRPORT_CRP	0	0.0
1301	ORIGIN_AIRPORT_CPR	0	0.0
1302	ORIGIN_AIRPORT_COU	0	0.0
1303	ORIGIN_AIRPORT_DRO	0	0.0
1304	DELAY_OR_NOT	0	0.0

1305 rows × 3 columns

In [21]:
```
'''
# step forward model selection
# gets the score of a given model, creates dict entry of model with its bic
def get_score(predictors):
    model = LogisticRegressionCV(
        Cs=list(np.power(10.0, np.arange(-10, 10)))
        ,penalty='l2'
        ,cv=5
        ,n_jobs=-1
        ,random_state=777
        ,fit_intercept=True
        ,solver='newton-cg',)
    model.fit(X_train[list(predictors)],y_train)
    return {"model": model, "score" : model.score(X_train, y_train)}

# determine the best of a given set of models
def best_of(predictors):
    remaining_predictors = [p for p in X_train.columns if p not in predictors]
    results = []
    for p in remaining_predictors:
        results.append(get_score(predictors+[p]))
    models = pd.DataFrame(results)
    best_model = models.loc[models['score'].argmax()]
    return best_model

models = pd.DataFrame(columns=["score", "model"])
predictors = []

# go through predictors stepwise until adding more predictors raises bic
for i in range(1, 651):
    models.loc[i] = best_of(predictors)
    predictors = models.loc[i]["model"].model.exog_names
    if i == 1:
        best_score = models.loc[i]["score"]
    else:
        if models.loc[i]["score"] < best_score:
            best_score = models.loc[i]["score"]
        if models.loc[i]["score"] > best_score:
            best_model = models.loc[i-1]
            print(best_model)
            break
'''
```

Out[21]:
```
'\n# step forward model selection\n# gets the score of a given model, creates di
ct entry of model with its bic\ndef get_score(predictors):\n    model = Logistic
RegressionCV(\n        Cs=list(np.power(10.0, np.arange(-10, 10)))\n        ,pen
alty=\'l2\'\n        ,cv=5\n        ,n_jobs=-1\n        ,random_state=777\n
   ,fit_intercept=True\n        ,solver=\'newton-cg\',)\n    model.fit(X_train[l
ist(predictors)],y_train)\n    return {"model": model, "score" : model.score(X_t
rain, y_train)}\n\n# determine the best of a given set of models\ndef best_of(pr
edictors):\n    remaining_predictors = [p for p in X_train.columns if p not in p
redictors]\n    results = []\n    for p in remaining_predictors:\n        result
s.append(get_score(predictors+[p]))\n    models = pd.DataFrame(results)\n    bes
t_model = models.loc[models[\'score\'].argmax()]\n    return best_model\n\nmodel
s = pd.DataFrame(columns=["score", "model"])\npredictors = []\n\n# go through pr
edictors stepwise until adding more predictors raises bic\nfor i in range(1, 65
1):\n    models.loc[i] = best_of(predictors) \n    predictors = models.loc[i]["m
```

```
odel"].model.exog_names\n    if i == 1:\n        best_score = models.loc[i]["sco
re"]        \n    else:\n        if models.loc[i]["score"] < best_score:\n
      best_score = models.loc[i]["score"]        \n        if models.loc[i]["sco
re"] > best_score:\n            best_model = models.loc[i-1]\n            print
(best_model)\n            break\n'
```

In [49]:
```
# Fit a logistic regression classifier with LASSO to the training set and report tl
clf = LogisticRegressionCV(
        Cs=list(np.power(10.0, np.arange(-10, 10)))
        ,penalty='l1'
        ,cv=5
        ,n_jobs=-1
        ,random_state=777
        ,fit_intercept=True
        ,solver='liblinear')
clf = clf.fit(X_train, y_train)


# Lasso Regularization parameter
print('\n')
print("The optimized L2 regularization paramater id:", clf.C_)

# The coefficients
print('Estimated beta1: \n', clf.coef_)
print('Estimated beta0: \n', clf.intercept_)

# Metrics
print('\n')
print('Test Set Confusion matrix:')
print(confusion_matrix(y_test, clf.predict(X_test)))
full_score = clf.score(X_full, y_full)
train_score = clf.score(X_train, y_train)
test_score = clf.score(X_test, y_test)
y_prediction = clf.predict(X_test)
test_precision = precision_score(y_test, y_prediction)
print('The training classification accuracy is: ', train_score)
print('The testing classification accuracy is: ', test_score)
print('The precision score on the test set is: ', test_precision)

print('The score on the full dataset is :', full_score )
```

```
C:\Users\wlt42\Anaconda3\lib\site-packages\sklearn\utils\validation.py:526: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Pl
ease change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)



The optimized L2 regularization paramater id: [ 1.]
Estimated beta1:
 [[  1.90520182e-01   8.17049374e-03  -2.37264328e-01 ...,   0.00000000e+00
    0.00000000e+00   9.13364944e+00]]
Estimated beta0:
 [-3.63728165]


Test Set Confusion matrix:
[[875   0]
 [  0 115]]
The training classification accuracy is:  1.0
The testing classification accuracy is:  1.0
The precision score on the test set is:  1.0
The score on the full dataset is : 0.999859091878
```

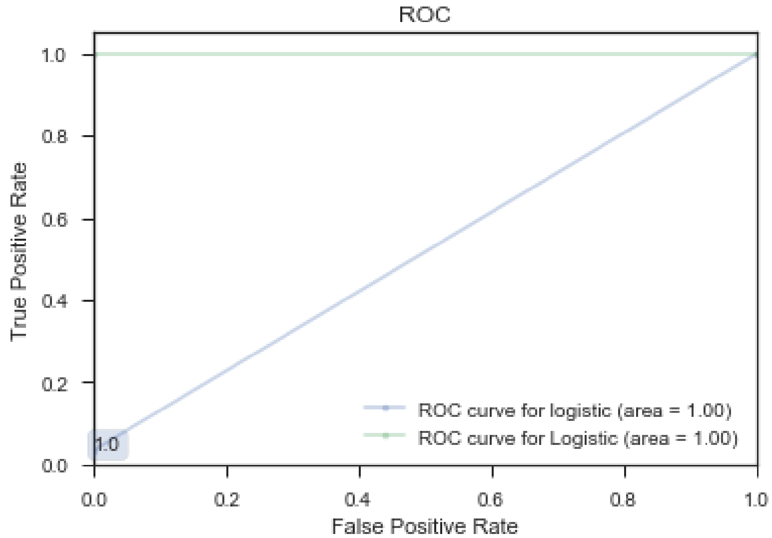
In [51]:

```
# from Lab, making ROC curves for this model, added ROC curve for our all negative
#manually making confusion table from a different threshold
def t_repredict(est, t, xtest):
    probs = est.predict_proba(xtest)
    p0 = probs[:,0]
    p1 = probs[:,1]
    ypred = (p1 > t)*1
    return ypred


from sklearn.metrics import roc_curve, auc

def make_roc(name, clf, ytest, xtest, ax=None, labe=5, proba=True, skip=0):
    initial=False
    if not ax:
        ax=plt.gca()
        initial=True
    if proba:#for stuff like logistic regression
        fpr, tpr, thresholds=roc_curve(ytest, clf.predict_proba(xtest)[:,1])
    else:#for stuff like SVM
        fpr, tpr, thresholds=roc_curve(ytest, clf.decision_function(xtest))
    roc_auc = auc(fpr, tpr)
    if skip:
        l=fpr.shape[0]
        ax.plot(fpr[0:l:skip], tpr[0:l:skip], '.-', alpha=0.3, label='ROC curve fo
    else:
        ax.plot(fpr, tpr, '.-', alpha=0.3, label='ROC curve for %s (area = %0.2f)'
    label_kwargs = {}
    label_kwargs['bbox'] = dict(
        boxstyle='round,pad=0.3', alpha=0.2,
    )
    if labe!=None:
        for k in range(0, fpr.shape[0],labe):
            #from https://gist.github.com/podshumok/c1d1c9394335d86255b8
            threshold = str(np.round(thresholds[k], 2))
            ax.annotate(threshold, (fpr[k], tpr[k]), **label_kwargs)
    if initial:
        ax.set_xlim([0.0, 1.0])
        ax.set_ylim([0.0, 1.05])
        ax.set_xlabel('False Positive Rate')
        ax.set_ylabel('True Positive Rate')
        ax.set_title('ROC')
    fpr_0, tpr_0, thresholds_1 = metrics.roc_curve(y_test, t_repredict(clf, 0.5, X
    roc_auc_0 = auc(fpr_0, tpr_0)
    plt.plot(fpr_0, tpr_0, '.-', alpha=0.3, label='ROC curve for Logistic (area = 
    ax.legend(loc="lower right")
    return ax

ax=make_roc("logistic",clf, y_test, X_test, labe=100, skip=2)
```



In [24]:
```
# QDA classification

clf = QuadraticDiscriminantAnalysis(store_covariances=True)
clf.fit(X_train, y_train.flatten(y_train.tolist()))

print ('The classifier had a Train score of:', clf.score(X_train,y_train))
print ('The classifier had a Test score of:', clf.score(X_test,y_test))
print('The confusion matrix is:')
print(confusion_matrix(y_test,clf.predict(X_test)))
#full_score = clf.score(X_full, y_full)
#print('The score on the full dataset is :', full_score )
```

```
C:\Users\wlt42\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DeprecationW
arning: Non-string object detected for the array ordering. Please pass in 'C',
 'F', 'A', or 'K' instead
  after removing the cwd from sys.path.
C:\Users\wlt42\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:695:
UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")

The classifier had a Train score of: 0.995024875622
The classifier had a Test score of: 0.883838383838
The confusion matrix is:
[[791  84]
 [ 31  84]]
```

In [52]:
```
# decision tree
from sklearn import tree
parameters = {'max_depth':range(2,10)}
clf = GridSearchCV(tree.DecisionTreeClassifier(), parameters, cv=5, n_jobs=4)
clf.fit(X_train, y_train.flatten(y_train.tolist()))
tree_model = clf.best_estimator_
print ('The classifier had the best CrossValidated score of:', clf.best_score_)
print('at a depth of:', clf.best_params_)
print ('The classifier had a Train score of:', tree_model.score(X_train,y_train))
print ('The classifier had a Test score of:', tree_model.score(X_test,y_test))
print('The confusion matrix is:')
print(confusion_matrix(y_test,tree_model.predict(X_test)))
full_score = clf.score(X_full, y_full)
print('The score on the full dataset is :', full_score )
print('The Confusion matrix on the full dataset:')
print(confusion_matrix(y_full,tree_model.predict(X_full)))
```

```
C:\Users\wlt42\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: DeprecationW
arning: Non-string object detected for the array ordering. Please pass in 'C',
 'F', 'A', or 'K' instead
  """

The classifier had the best CrossValidated score of: 1.0
at a depth of: {'max_depth': 2}
The classifier had a Train score of: 1.0
The classifier had a Test score of: 1.0
The confusion matrix is:
[[875   0]
 [  0 115]]
The score on the full dataset is : 1.0
The Confusion matrix on the full dataset:
[[720168      0]
 [     0  81773]]
```

In [53]:
```
plt.plot(flights_sample_2.AIRLINE[:990], tree_model.predict(X_test), color="blue",
#plt.plot(k_list, rtest_list, color="red", label='test R^2')
plt.xlabel('AIRLINE')
plt.ylabel('DELAYS')
plt.title('Predicted Delays by AIRLINE')
plt.axis('tight')
plt.legend(loc='best')

plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-53-1f03682c37b3> in <module>()
----> 1 plt.plot(flights_sample_2.AIRLINE[:990], tree_model.predict(X_test), col
or="blue", label='Predicted Delays by AIRLINE')
      2 #plt.plot(k_list, rtest_list, color="red", label='test R^2')
      3 plt.xlabel('AIRLINE')
      4 plt.ylabel('DELAYS')
      5 plt.title('Predicted Delays by AIRLINE')

C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\pyplot.py in plot(*args, *
*kwargs)
   3315                         mplDeprecation)
   3316     try:
-> 3317         ret = ax.plot(*args, **kwargs)
   3318     finally:
   3319         ax._hold = washold

C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\__init__.py in inner(ax, *
args, **kwargs)
   1895                     warnings.warn(msg % (label_namer, func.__name__),
   1896                                   RuntimeWarning, stacklevel=2)
-> 1897             return func(ax, *args, **kwargs)
   1898         pre_doc = inner.__doc__
   1899         if pre_doc is None:

C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py in plot(sel
f, *args, **kwargs)
   1405 
   1406         for line in self._get_lines(*args, **kwargs):
-> 1407             self.add_line(line)
   1408             lines.append(line)
   1409 

C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in
add_line(self, line)
   1791             line.set_clip_path(self.patch)
   1792 
-> 1793         self._update_line_limits(line)
   1794         if not line.get_label():
   1795             line.set_label('_line%d' % len(self.lines))

C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in _update_l
ine_limits(self, line)
   1813         Figures out the data limit of the given line, updating self.data
Lim.
   1814         """
```

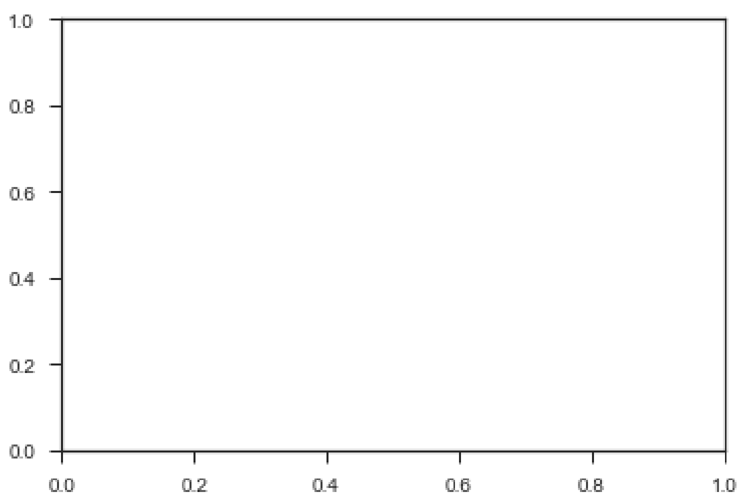
```
-> 1815         path = line.get_path()
   1816         if path.vertices.size == 0:
   1817             return

C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\lines.py in get_path(self)
    987         """
    988         if self._invalidy or self._invalidx:
--> 989             self.recache()
    990         return self._path
    991 

C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\lines.py in recache(self, always)
    674                 x = ma.asarray(xconv, np.float_).filled(np.nan)
    675             else:
--> 676                 x = np.asarray(xconv, np.float_)
    677             x = x.ravel()
    678         else:

C:\Users\wlt42\Anaconda3\lib\site-packages\numpy\core\numeric.py in asarray(a, dtype, order)
    529 
    530     """
--> 531     return array(a, dtype, copy=False, order=order)
    532 
    533 

ValueError: could not convert string to float: 'OO'
```



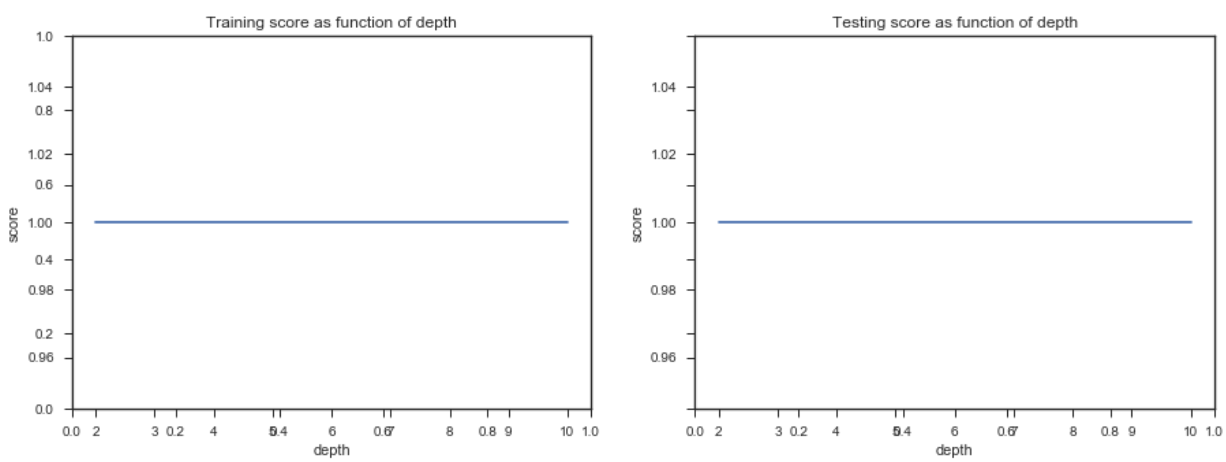
In [27]:
```
# decision tree depth visualization
depths = [2,3,4,5,6,7,8,9,10]
fig, ax= plt.subplots(1, 2, figsize=(15, 5), sharey=True)
scores_train = []
scores_test = []
for depth in depths:
    dt = tree.DecisionTreeClassifier(max_depth = depth)
    dt.fit(X_train, y_train)
    scores_train.append(dt.score(X_train, y_train))
ax5 = fig.add_subplot(121)
ax6 = fig.add_subplot(122)
ax5.set_xlabel('depth')
ax5.set_ylabel('score')
ax5.set_title('Training score as function of depth')
ax5.legend()
ax5.plot(depths, scores_train)

for depth in depths:
    dt = tree.DecisionTreeClassifier(max_depth = depth)
    dt.fit(X_train, y_train)
    scores_test.append(dt.score(X_test, y_test))

ax6.set_xlabel('depth')
ax6.set_ylabel('score')
ax6.set_title('Testing score as function of depth')
ax6.legend()
ax6.plot(depths, scores_test)
```

```
C:\Users\wlt42\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:545: UserWarning: No labelled objects found. Use label='...' kwarg on individual plots.
  warnings.warn("No labelled objects found. "
```

Out[27]: [<matplotlib.lines.Line2D at 0x1a4804765f8>]



In [ ]:
```
flights.describe()
```

In [28]:
```
# drop unecessary columns and seperate X and y
#y = ((np.array(flights_sample['DELAY_OR_NOT'])))
y_full = flights_2['ARRIVAL_DELAY'].values
y = flights_sample_2['ARRIVAL_DELAY'].values
flights_sample_2 = flights_sample_2.drop(['ARRIVAL_DELAY','DATE','AIRLINE','SCHED_
flights_2 = flights_2.drop(['ARRIVAL_DELAY','DATE','AIRLINE','SCHED_DEP','SCHED_AR
X = flights_sample_2.loc[:].values
X_full = flights_2.loc[:].values

# split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_s
```

In [29]:
```
# check shape of data
y_train = y_train.reshape(len(y_train), 1)
y_test = y_test.reshape(len(y_test), 1)
print(np.shape(X_train), np.shape(y_train), np.shape(X_test), np.shape(y_test))
```

```
(2010, 1304) (2010, 1) (990, 1304) (990, 1)
```

In [30]:
```
from sklearn.linear_model import LassoCV
lasso = LassoCV(cv=4)
lasso.fit(X_train, y_train)
y_pred = lasso.predict(X_test)

print('The equation of the regression plane is: {} + {}^T . x'.format(lasso.interc
train_MSE= np.mean((y_train - lasso.predict(X_train))**2)
test_MSE= np.mean((y_test - lasso.predict(X_test))**2)

print('The train MSE is {}, the test MSE is {}'.format(train_MSE, test_MSE))

train_R_sq = lasso.score(X_train, y_train)
test_R_sq = lasso.score(X_test, y_test)
test_R_sq_full = lasso.score(X_full, y_full)
print('The train R^2 is {}, the test R^2 is {}'.format(train_R_sq, test_R_sq))
print('The R2 on full dataset:', test_R_sq_full )
```

```
C:\Users\wlt42\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_desce
nt.py:1082: DataConversionWarning: A column-vector y was passed when a 1d array
 was expected. Please change the shape of y to (n_samples, ), for example using
 ravel().
  y = column_or_1d(y, warn=True)

The equation of the regression plane is: -0.17742450437283325 + [  9.98922036e-0
1  -4.94188541e-04  -9.82338633e-01 ...,  -0.00000000e+00
   0.00000000e+00  -0.00000000e+00]^T . x
The train MSE is 2118.190601238928, the test MSE is 1876.3183037572326
The train R^2 is 0.9999711219063213, the test R^2 is 0.9999629104122523
The R2 on full dataset: 0.999964460173
```

We will build two separate models: one model that classifies whether a flight will be delayed and a second model that predicts the length of delay given that a flight is truly delayed. Only consider models taught in class so far.

Consider the following: This is a large dataset; think of strategies on how to solve this problem. Create a manageable subsample of the data that you can use to train and test/validate, but eventually you should predict on all the data (excluding the training set).

Questions

1. (5pts) Create a new variable, `DELAY_OR_NOT`: a boolean/indicator variable which indicates any arrival delay under 15 mins as a 0, and any delay at or above 15 mins as a 1 (`ARRIVAL_DELAY >= 15`).
2. (5pts) Make sure you understand the data variable descriptions before you start the analysis. Consider all the columns and determine and list which of these predictors should not be used.
3. (15pts) Perform EDA to gain intuition of the factors that affect delay and provide visuals: do delays vary across airlines, or time of departure, or airport (do, at the very least, Chicago (ORD), Boston (BOS), and your favorite another airport), or airport traffic?
4. (20pts) Build a classification model that classifies delays according to `DELAY_OR_NOT`. This is an unbalanced dataset, thus consider the appropriate performance metric when reporting your results.
5. (5pts) Given your model, comment on the importance of factors as related to whether a flight is delayed.
6. (5pts) Evaluate your model(s) on your test set, and finally provide a visual to show which airlines are predicted to have the most delays using all the data excluding the training and test set.
7. (15pts) Build a regression model that predicts the length of delay (on the log scale) given that a flight is truly delayed.
8. (20pts) Write a report (in the last markdown cell in your notebook with your findings (without code)). Describe the main design decisions you have made with justifications. Clearly explain your methodology and results. This should not be more than 300 words. You may use up to 5 diagrams.

In [ ]:
```
# statsmodel regression
# create the X matrix by appending a column of ones to x_train
X = sm.add_constant(X_train)
X_test_sm = sm.add_constant(X_test)
# build the OLS model from the training data
smm = sm.OLS(y_train, X)

#save regression info in results_sm
results_sm = smm.fit()

print(results_sm)
print(results_sm.summary())
print('Parameters: ', results_sm.params)
```

209 Additional questions

1. (10pts) Engineer two additional features that will help improve the classification model's performance.
2. (5pts) Add one additional feature from a data source not given to you. Do this only after you complete the rest of the exam.

Deliverable:

A well presented notebook with well structured and documented code to answer questions 1-7 (plus additional questions for 209 students) with brief explanations and/or clarifications (10pts for overall presenation). The last cell should contain the report for question 8.

Hints

1. For the classification model, an AUC of approximately 0.6 should be your base model.
2. $R^2 > 0.03$ for the regression is good, $R^2 > 0.05$ very good, and $R^2 > 0.1$ is impressive (measured on the log scale).

```
I first created a sample of n=3000 to explore the data. I additionally kept the
full dataset and made the same changes on it as the sample set. I removed the
following predictors:
'ARRIVAL_DELAY','DATE','AIRLINE','SCHED_DEP','SCHED_ARR','FLIGHT_NUMBER','TAIL_N
UMBER','AIR_SYSTEM_DELAY','SECURITY_DELAY','AIRLINE_DELAY','LATE_AIRCRAFT_DELAY'
,'WEATHER_DELAY'.
Date duplicates information already accounted for by monthe, day, and day of the
wek. Tail number and flight number are nomianal and contribute nothing to our
model. All of the DELAYS were removed as well, as these just indicate a reason
for a delay. Additionally the data in these columns was missing upwards of 90%
of the values. I removed SCHED_ARR and SCHED_DEP after onehot encoding each hour
of the day. This made my model unnecessarily complex, so I abandoned this line
after of reasoning after seeing scores for some other models.
I removed AIRLIN under the assumption that much of commercial airport traffic is
highly regulated and thus delays may result more from airport conditions rather
than any particular airline. Also, my graph showed not an unreasonable
difference in mean delays. I realize this may be a faulty assumption, but the
models performance eased this worry. Although perhaps relevent to a flights
ontime status, the other predictors clearly made up for this. I implemented a
cross validated decision tree,  tuning for depth, a QDA, and a cross validated
Lasso Logistic Regression, tuning for alpha.

The best results came from the decision tree with:
The classifier had the best CrossValidated score of: 1.0
at a depth of: {'max_depth': 2}
The classifier had a Train score of: 1.0
The classifier had a Test score of: 1.0
The confusion matrix is:
[[875   0]
 [  0 115]]
The score on the full dataset is : 1.0
The Confusion matrix on the full dataset:
[[722834      0]
 [     0  82107]]

 I saw no need to fit an ROC curve, as the classifier made no errors on the full
dataset. Also, the cost of a false negative and a false postive seem equivelant.
This model beat logistic regression by a fraction of a percent. Having had these
results on the entire dataset, I was happy with the results. Though the models
is complext with high dimensionality, its performance is superb, and the runtime
was minimal.
```

For the regression, I performed an OLS regression, so that I could see p values easily. This with the notion to remove excess predictors. At the same time, I fit a cross validated Lasso linear regression, tuning for alpha. The results again were impressive:
The train R^2 is 0.9999711219063213, the test R^2 is 0.9999629104122523
The R2 on full dataset: 0.999964478576

As far as inference, the 6th through the 51st predictors had the biggets impact. This range is the onehot encodings of month, day and day oth the week. Othe predictors certainly had an impact, especially certain arrival airports, but the date and time had the biggest impact.

Additional things I could have done. Fully implemented stepwise predictor selection. The models seems like they could get much simpler, yet there performance was near perfect or perfect.