Import Appointment Data

In [1]:

```
import pandas as pd
import numpy as np
#Load Data Section
data = pd.read_csv('./KaggleV2-May-2016.csv',parse_dates=['ScheduledDay'
, 'AppointmentDay'])
data.head()
```

Out[1]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbo
0	2.987250e+13	5642903	F	2016-04-29 18:38:08	2016-04-29	62	JARDIM PENHA
1	5.589978e+14	5642503	M	2016-04-29 16:08:27	2016-04-29	56	JARDIM PENHA
2	4.262962e+12	5642549	F	2016-04-29 16:19:04	2016-04-29	62	MATA DA
3	8.679512e+11	5642828	F	2016-04-29 17:29:31	2016-04-29	8	PONTAL CAMBUR
4	8.841186e+12	5642494	F	2016-04-29 16:07:23	2016-04-29	56	JARDIM PENHA

MungeData

In [2]:

```
#Munge Data
from sklearn.preprocessing import LabelEncoder
#Day of week, eg. Friday or Monday may be good predictor
data['ScheduleDayOfWeek'] = data.ScheduledDay.dt.dayofweek
data['AppointmentDayOfWeek'] = data.AppointmentDay.dt.dayofweek
data['SameDayAppt'] = np.where(data.ScheduledDay.dt.dayofweek == data.Ap
pointmentDay.dt.dayofweek,1,0)
data['HoursUntilAppt'] = (data.AppointmentDay-data.ScheduledDay)
#convert yes, no to 0, 1
data['No-show-binary'] = np.where(data['No-show'] == 'Yes',1,0)
data['Gender-binary'] = np.where(data['Gender'] == 'Yes',1,0)

#convert neighbourhood into dummy columns
dummies = pd.get_dummies(data.Neighbourhood)
data[dummies.columns] = dummies
data.head()
```

Out[2]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbo
0	2.987250e+13	5642903	F	2016-04-29 18:38:08	2016-04-29	62	JARDIM PENHA
1	5.589978e+14	5642503	M	2016-04-29 16:08:27	2016-04-29	56	JARDIM PENHA
2	4.262962e+12	5642549	F	2016-04-29 16:19:04	2016-04-29	62	MATA DA
3	8.679512e+11	5642828	F	2016-04-29 17:29:31	2016-04-29	8	PONTAL CAMBUR
4	8.841186e+12	5642494	F	2016-04-29 16:07:23	2016-04-29	56	JARDIM PENHA

5 rows × 101 columns

Data Analysis

In [3]:

```
num_columns = [col for col, dtype
    in zip(data.columns, data.dtypes) if dtype in ['float64', 'int64',
'uint8'] and
    col not in ['No-show-binary','PatientId', 'AppointmentID'] ]

X = data[num_columns]
y = data['No-show-binary']
```

In [4]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

logistic = LogisticRegression()
kn = KNeighborsClassifier()

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25
)

logistic.fit(x_train, y_train)
kn.fit(x_train, y_train)
```

Out[4]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

Test Models

In [5]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classifica
tion_report

#Logistic Regression
print('logistic accuracy score: ' + str(accuracy_score(y_test, logistic.
predict(x_test))))
print('logistic confusion matrix: ' + str(confusion_matrix(y_test, logis
tic.predict(x_test))))
print('logistic classification report')
print(classification_report(y_test, logistic.predict(x_test)))
```

```
logistic accuracy score: 0.796612623046
logistic confusion matrix: [[22006    10]
 [ 5610     6]]
logistic classification report
             precision    recall  f1-score   support

          0       0.80      1.00      0.89     22016
          1       0.38      0.00      0.00      5616

avg / total       0.71      0.80      0.71     27632
```

```
In [6]: #KNearest Neighbors
        print('k-nearest neighbor accuracy score: ' + str(accuracy_score(y_test,
         kn.predict(x_test))))
        print('k-nearest neighbor confusion matrix: ' + str(confusion_matrix(y_t
        est, kn.predict(x_test))))
        print('k-nearest classification report')
        print(classification_report(y_test, kn.predict(x_test)))
```

```
k-nearest neighbor accuracy score: 0.772510133179
k-nearest neighbor confusion matrix: [[20397  1619]
 [ 4667   949]]
k-nearest classification report
             precision    recall  f1-score   support

          0       0.81      0.93      0.87     22016
          1       0.37      0.17      0.23      5616

avg / total       0.72      0.77      0.74     27632
```

Try subset of data

Remove neighbourhood dummy columns

```
In [7]: reduced_set = num_columns[0:11]
        X = data[reduced_set]
        x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25
        )

        logistic.fit(x_train, y_train)
        kn.fit(x_train, y_train)
```

```
Out[7]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
        i',
                   metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                   weights='uniform')
```

In [8]:

```
print('logistic accuracy score: ' + str(accuracy_score(y_test, logistic.
predict(x_test))))
print('logistic confusion matrix: ' + str(confusion_matrix(y_test, logis
tic.predict(x_test))))
print('logistic classification report')
print(classification_report(y_test, logistic.predict(x_test)))
```

```
logistic accuracy score: 0.798204979734
logistic confusion matrix: [[22056     0]
 [ 5576     0]]
logistic classification report
             precision    recall  f1-score   support

          0       0.80      1.00      0.89     22056
          1       0.00      0.00      0.00      5576

avg / total       0.64      0.80      0.71     27632


/Users/jamescheever/anaconda3/lib/python3.6/site-packages/sklearn/metri
cs/classification.py:1135: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samp
les.
  'precision', 'predicted', average, warn_for)
```

In [9]:

```
print('k-nearest neighbor accuracy score: ' + str(accuracy_score(y_test,
 kn.predict(x_test))))
print('k-nearest neighbor confusion matrix: ' + str(confusion_matrix(y_t
est, kn.predict(x_test))))
print('k-nearest classification report')
print(classification_report(y_test, kn.predict(x_test)))
```

```
k-nearest neighbor accuracy score: 0.764837869137
k-nearest neighbor confusion matrix: [[20286  1770]
 [ 4728   848]]
k-nearest classification report
             precision    recall  f1-score   support

          0       0.81      0.92      0.86     22056
          1       0.32      0.15      0.21      5576

avg / total       0.71      0.76      0.73     27632
```

```
In [10]: X = data[['Age']]
         x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25
         )
         logistic.fit(x_train, y_train)
         print('logistic accuracy score: ' + str(accuracy_score(y_test, logistic.
         predict(x_test))))
         print('logistic confusion matrix: ' + str(confusion_matrix(y_test, logis
         tic.predict(x_test))))
         print('logistic classification report')
         print(classification_report(y_test, logistic.predict(x_test)))
```

```
logistic accuracy score: 0.797806890562
logistic confusion matrix: [[22045     0]
 [ 5587     0]]
logistic classification report
             precision    recall  f1-score   support

          0       0.80      1.00      0.89     22045
          1       0.00      0.00      0.00      5587

avg / total       0.64      0.80      0.71     27632

/Users/jamescheever/anaconda3/lib/python3.6/site-packages/sklearn/metri
cs/classification.py:1135: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samp
les.
  'precision', 'predicted', average, warn_for)
```