In [1]:
```
# Because we learned in our data cleaning, exploratory data analysis, and data visualiz
# Shelter Data were of poor quality, we decided to determine whether - if the shelter h
# a Logistic Regression model could predict outcome.

# For this model, we defined outcome as "Placement"= 1 (either adopted or returned to o
# "Placement" = 0 (all other outcomes -died, euthanasia, transferred, etc.)  So Placeme
# was placed in a home, whether its previous home or a new home.

# To assess whether such a model could be fit if the shelter had more accurate data, we
# that had one or more clearly erroneous data points.  These included missing dates of
# intake to the shelter, negative lengths of stay at the shelter, animals who were spay
# but no longer spayed/neutered at outcome, unknown sex/age, etc.

# Since only 4 animals classified as Intake Type = Wildlife were placed in a home, this
# predictive model.  Therefore, we eliminated the Wildlife category for this analysis.
# Type = Euthanasia Request, which accounted for 0.2% of all intakes; this category and
# eliminated.

# After this cleaning, 108,399 observations remained.  This is a significant decrease f
# observations but still a large data set.
```

In [2]:
```
import pandas as pd
%matplotlib inline
import numpy as np
import seaborn as sb
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import sklearn
```

In [3]:
```
from pandas import Series, DataFrame
from pylab import rcParams
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict
from sklearn import metrics
```

In [4]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score
rcParams['figure.figsize']=5,4
sb.set_style('whitegrid')
```

In [5]:
```
LRData_df = pd.read_csv("Austin_Animal_Shelter_Dataset_LR.csv")

LRData_df.head()
LRData_df[0:10]
LRData_df.isnull().any()
print(LRData_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108399 entries, 0 to 108398
Data columns (total 16 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Intake Age in Months  108399 non-null  int64
 1   Abandoned             108399 non-null  int64
 2   Owner Surrender       108399 non-null  int64
 3   Public Assist         108399 non-null  int64
```

```
 4   Stray                108399 non-null  int64
 5   Normal               108399 non-null  int64
 6   SickInj              108399 non-null  int64
 7   PregNurs             108399 non-null  int64
 8   Feral                108399 non-null  int64
 9   Bird                 108399 non-null  int64
 10  Cat                  108399 non-null  int64
 11  Dog                  108399 non-null  int64
 12  Livestock            108399 non-null  int64
 13  Other                108399 non-null  int64
 14  Male                 108399 non-null  int64
 15  Placement            108399 non-null  int64
dtypes: int64(16)
memory usage: 13.2 MB
None
```

In [6]:
```
# We will use 75% of the data as a Training Set and 25% as a Test Set.
# Placement is the Target Variable.

x_train, x_test, y_train, y_test = train_test_split(LRData_df.drop('Placement',axis=1),

# Check Number of Observations and Number of Columns:

print(x_train.shape)
print(y_train.shape)

# Preview a few rows of the Training Set:
x_train[0:5]
```

```
(81299, 15)
(81299,)
```

Out[6]:

	Intake Age in Months	Abandoned	Owner Surrender	Public Assist	Stray	Normal	SickInj	PregNurs	Feral	Bird	Cat	Dc
35540	36	0	0	0	1	0	1	0	0	0	0	
22727	10	0	1	0	0	1	0	0	0	0	0	
71185	0	0	0	0	1	1	0	0	0	0	1	
30296	0	0	0	0	1	1	0	0	0	0	0	
47999	24	0	0	0	1	1	0	0	0	0	0	

In [7]:
```
LogReg = LogisticRegression(solver='liblinear')
LogReg.fit(x_train, y_train)

y_pred = LogReg.predict(x_test)
```

In [8]:
```
# Print Classification Report to look at model accuracy

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.62      0.14      0.22      9071
           1       0.69      0.96      0.80     18029

    accuracy                           0.68     27100
   macro avg       0.65      0.55      0.51     27100
```

```
weighted avg       0.66      0.68      0.61     27100
```

In [9]:
```
# Generate a Confusion Matrix for another look at accuracy

y_train_pred = cross_val_predict(LogReg, x_train, y_train, cv=5)
confusion_matrix(y_train, y_train_pred)
precision_score(y_train, y_train_pred)
```

Out[9]:
```
0.6832311412121455
```

In [10]:
```
# The Logistic Regression above including all animal types did not have excellent resul
# the most typical house pets, Cats and Dogs.  As before, we eliminated all Intake Type
# which accounted for 0.2% of all intakes.

# After this cleaning, 107,064 observations remained.  This is a significant decrease f
# observations but still a large data set and nearly as many as the set with all Animal
```

In [11]:
```
LRCatDogData_df = pd.read_csv("Austin_Animal_Shelter_Dataset_LR_CatDog.csv")

LRCatDogData_df.head()
LRCatDogData_df[0:10]
LRCatDogData_df.isnull().any()
print(LRCatDogData_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 107064 entries, 0 to 107063
Data columns (total 13 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Intake Age in Months  107064 non-null  int64
 1   Abandoned             107064 non-null  int64
 2   Owner Surrender       107064 non-null  int64
 3   Public Assist         107064 non-null  int64
 4   Stray                 107064 non-null  int64
 5   Normal                107064 non-null  int64
 6   SickInj               107064 non-null  int64
 7   PregNurs              107064 non-null  int64
 8   Feral                 107064 non-null  int64
 9   Cat                   107064 non-null  int64
 10  Dog                   107064 non-null  int64
 11  Male                  107064 non-null  int64
 12  Placement             107064 non-null  int64
dtypes: int64(13)
memory usage: 10.6 MB
None
```

In [12]:
```
# We will use 75% of the data as a Training Set and 25% as a Test Set.
# Placement is the Target Variable.

x_train, x_test, y_train, y_test = train_test_split(LRCatDogData_df.drop('Placement',ax

# Check Number of Observations and Number of Columns:

print(x_train.shape)
print(y_train.shape)

# Preview a few rows of the Training Set:
x_train[0:5]
```

```
(80298, 12)
(80298,)
```

Out[12]:

	Intake Age in Months	Abandoned	Owner Surrender	Public Assist	Stray	Normal	SickInj	PregNurs	Feral	Cat	Dog	Ma
22971	60	0	0	1	0	1	0	0	0	0	1	
48218	24	0	1	0	0	1	0	0	0	0	1	
3326	41	0	0	0	1	1	0	0	0	0	1	
58304	0	0	0	0	1	1	0	0	0	1	0	
99320	2	0	0	0	1	1	0	0	0	0	1	

In [13]:
```
LogReg = LogisticRegression(solver='liblinear')
LogReg.fit(x_train, y_train)

y_pred = LogReg.predict(x_test)
```

In [14]:
```
# Print Classification Report to look at model accuracy

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.60      0.14      0.23      8976
           1       0.69      0.95      0.80     17790

    accuracy                           0.68     26766
   macro avg       0.65      0.55      0.51     26766
weighted avg       0.66      0.68      0.61     26766
```

In [15]:
```
# Generate a Confusion Matrix for another look at accuracy

y_train_pred = cross_val_predict(LogReg, x_train, y_train, cv=5)
confusion_matrix(y_train, y_train_pred)
precision_score(y_train, y_train_pred)
```

Out[15]:
```
0.6869179120641109
```

In [16]:
```
# The Model was no more accurate when limited to cats and dogs.
```