DATA ANALYTICS PROJECT:

Group members: Jesse Reisner, Ryan Cree, Morgan McKee, Melanie Klock

Introduction:
This analysis is focused on using features of animals at the Austin Animal Shelter to predict the outcome for that aniaml (adoption, transfer, euthansia, etc.). This problem matters because we could identify which animals are more "at risk" for being put to sleep in shelters and focus specialized measures to get these animals adopted. We can also identify what traits of animals (breed, color, age) correlates with being adopted. These results could be used to educate shelters on what animals are being adopted more often and help shelters predict if an incoming animal is more at danger for never being adopted. These results could also be used to educate the general public on what breeds are often overlooked for adoption and show that certain animals need a home!

The dataset we are using to solve this problem is from the Austin Animal Shelter. It has about 65k records of a variety of animals that have been at the shelter, as well as traits of these animals including species, breed, name, age, coloring, etc. We will use this dataset to build models that can predict, based on an animal's characteristic, its probability of being adopted.

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import time

sns.set()
```

In [8]:
```
#read data:
data = pd.read_csv('Austin_Animal_Center_Outcomes.csv')
data.head()
```

Out[8]:

	Animal ID	Name	DateTime	MonthYear	Date of Birth	Outcome Type	Outcome Subtype	Animal Type	Se upo Outcom
0	A794011	Chunk	05/08/2019 06:20:00 PM	05/08/2019 06:20:00 PM	05/02/2017	Rto-Adopt	NaN	Cat	Neutere Mal
1	A776359	Gizmo	07/18/2018 04:02:00 PM	07/18/2018 04:02:00 PM	07/12/2017	Adoption	NaN	Dog	Neutere Mal
2	A720371	Moose	02/13/2016 05:59:00 PM	02/13/2016 05:59:00 PM	10/08/2015	Adoption	NaN	Dog	Neutere Mal
3	A674754	NaN	03/18/2014 11:47:00 AM	03/18/2014 11:47:00 AM	03/12/2014	Transfer	Partner	Cat	Intac Mal
4	A689724	*Donatello	10/18/2014 06:52:00 PM	10/18/2014 06:52:00 PM	08/01/2014	Adoption	NaN	Cat	Neutere Mal

In [9]:
```
data.shape
```

Out[9]: (116560, 12)

DATA PREP:

Overview: DateTime is a column identical to MonthYear and therefore can be excluded. We also removed rows that do not have a class label. Outcome Subtype column also has mostly NAs, and only provide more detail to the class label, but wouldn't be needed for the model, so we can exclude that as well. Finally, we fixed the Age column to be numerical, in days, for simplicity.

In [10]:
```
#drop rows without class label
print(data['Outcome Type'].isna().sum())
data = data.dropna(subset = ['Outcome Type'])
print(data.shape)
```

```
6
(116554, 12)
```

In [11]:
```
data['Outcome Subtype'].isna().sum()
```

Out[11]: 63914

In [12]:
```
#checking each column for any NaN values
for column in data.columns:
    print(column, ": ", data[column].isna().sum())
```

```
Animal ID :  0
Name :  36481
DateTime :  0
MonthYear :  0
Date of Birth :  0
Outcome Type :  0
Outcome Subtype :  63914
Animal Type :  0
Sex upon Outcome :  2
Age upon Outcome :  37
Breed :  0
Color :  0
```

Outcome Subtype has more than half NaN values, so we will exclude this column from the dataset. For the other columns with missing values, we will drop those rows, since there aren't too many of them. Additionally, outcome subtype only adds additional information on the feature outcome.

Name also has quite a few NaN values, but later on we can use this column for data exploration and feature engineering, so we will keep it for now. AnimalID is the key for this table, so it will also not be used in the model but could be useful for indexing, so we'll also keep it for now.

In [13]:
```
#drop columns we don't need:
data = data.drop(columns = ['Outcome Subtype', 'DateTime'])
data.shape
```

Out[13]: (116554, 10)

In [14]:
```
#drop rows with missing data:
data = data.dropna(subset = ['Sex upon Outcome', 'Age upon Outcome'])
data.shape
```

Out[14]: (116516, 10)

In [15]:
```
#fill NAs for Name column:
data['Name'] = data['Name'].fillna(value = "None")
```

We'll also make the MonthYear column a datetime object.

In [16]:
```
data['MonthYear'] = data['MonthYear'].astype('datetime64[ns]')
```

Next, we'll make the Age upon Outcome column numerical by transforming everything into days. There are some negative numbers, which are errors, and we'll drop these from our dataset.

There are some negative values, and also quite a few '0 years'. Since these values won't translate to days' units that make sense, we'll remove these rows.

In [17]:
```
#this function takes a length of time and transforms it into # of days.
#If there is a negative # , a zero, or the input does not match any formats, g
ive the cell a None value.
# we will say one month has 30 days.
def time_days(time):
    timesplit = time.split()
    number = timesplit[0]
    if(int(number) < 0): #if number is negative, return None
        return None
    unit = timesplit[1]
    if 'year' in unit:
        return (int(number) * 365)
    elif 'month' in unit:
        return (int(number) * 30)
    elif 'week' in unit:
        return (int(number) * 7)
    elif 'days' in unit:
        return (int(number))
```

In [18]:
```
#age --> days
data['Age upon Outcome'] = data.apply(lambda x: time_days(x['Age upon Outcome'
]), axis = 1)
```

In [19]:
```
data.head()
```

Out[19]:

	Animal ID	Name	MonthYear	Date of Birth	Outcome Type	Animal Type	Sex upon Outcome	Age upon Outcome	
0	A794011	Chunk	2019-05-08 18:20:00	05/02/2017	Rto-Adopt	Cat	Neutered Male	730.0	Domestic
1	A776359	Gizmo	2018-07-18 16:02:00	07/12/2017	Adoption	Dog	Neutered Male	365.0	(Sh
2	A720371	Moose	2016-02-13 17:59:00	10/08/2015	Adoption	Dog	Neutered Male	120.0	Shepherc
3	A674754	None	2014-03-18 11:47:00	03/12/2014	Transfer	Cat	Intact Male	6.0	Domestic
4	A689724	*Donatello	2014-10-18 18:52:00	08/01/2014	Adoption	Cat	Neutered Male	60.0	Domestic

In [20]:
```
#drop rows with missing ages:
print(data['Age upon Outcome'].isna().sum())
data = data.dropna(subset = ['Age upon Outcome'])
data.shape
```

```
245
```

Out[20]:
```
(116271, 10)
```

One last step. The column names are a bit long and have spaces, so we'll shorten them for efficiency.

In [21]:
```
#change columns
clean_data = data.copy() #make a copy of the data for EDA
clean_data.columns = ['id', 'name', 'date', 'dob', 'outcome', 'animal_type','s
ex', 'age', 'breed', 'color']
clean_data.head()
```

Out[21]:

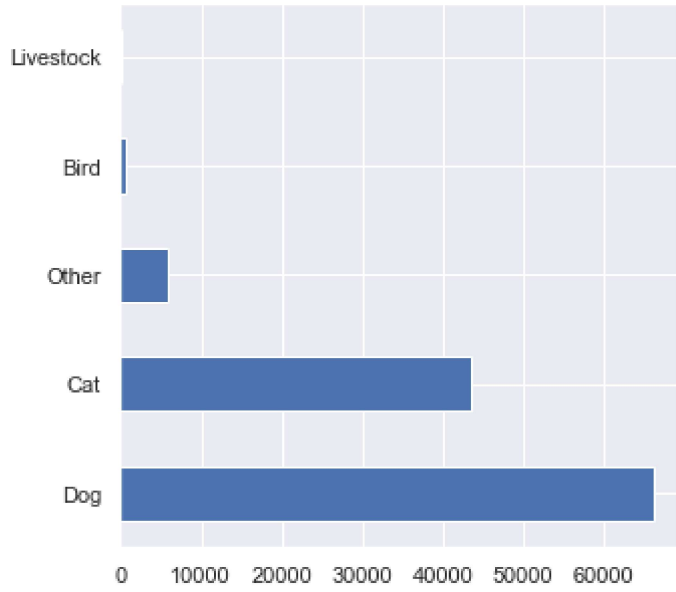
	id	name	date	dob	outcome	animal_type	sex	age	
0	A794011	Chunk	2019-05-08 18:20:00	05/02/2017	Rto-Adopt	Cat	Neutered Male	730.0	Domestic
1	A776359	Gizmo	2018-07-18 16:02:00	07/12/2017	Adoption	Dog	Neutered Male	365.0	C Sho
2	A720371	Moose	2016-02-13 17:59:00	10/08/2015	Adoption	Dog	Neutered Male	120.0	Shepherd/
3	A674754	None	2014-03-18 11:47:00	03/12/2014	Transfer	Cat	Intact Male	6.0	Domestic
4	A689724	*Donatello	2014-10-18 18:52:00	08/01/2014	Adoption	Cat	Neutered Male	60.0	Domestic

DATA EXPLORATION:

Overview:

In [22]:
```
#records graphed by animal type
clean_data['animal_type'].value_counts().plot(kind = 'barh', figsize=(5,5))
```

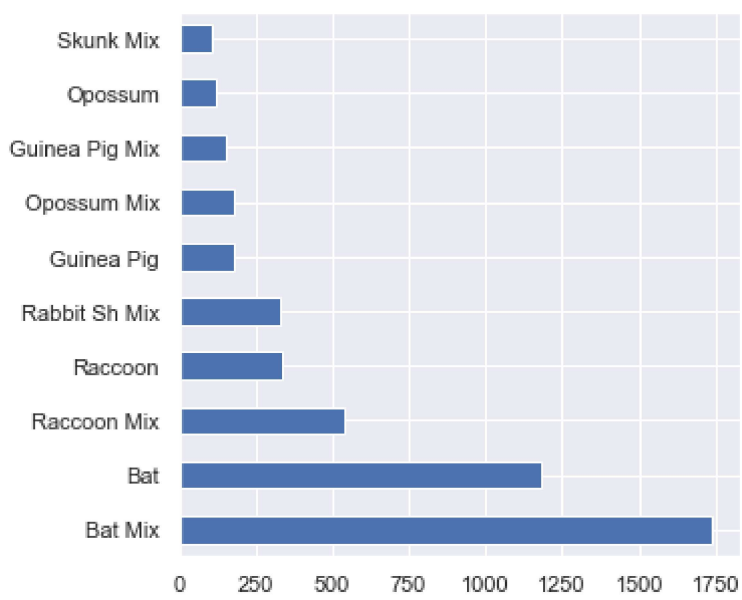
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1dccc1d0>



We can see here that there are mostly dogs and cats in the shelter. Surprisingly, there are also birds and livestock!

In [23]:
```
#records graphed by animal type
clean_data[clean_data.animal_type == 'Other'].breed.value_counts().iloc[:10].p
lot(kind = 'barh', figsize=(5,5))
```

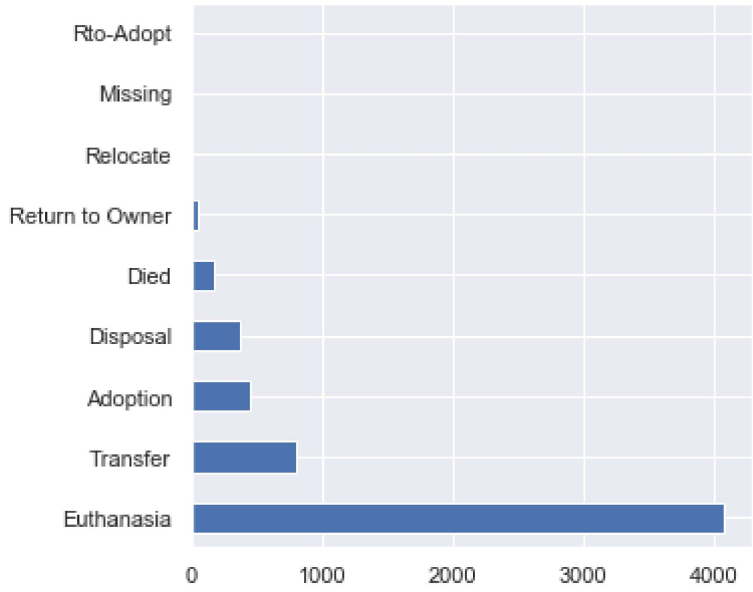
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1a229dcfd0>



For the "Other" value for animal_type, it looks like these mostly include bats or raccoons. Interesting that these are included in the dataset for the shelter as well. These animals aren't adopted normally, so what happens to them?

In [24]:
```
clean_data[clean_data.animal_type == 'Other'].outcome.value_counts().iloc[:10]
.plot(kind = 'barh', figsize=(5,5))
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a22986810>
```

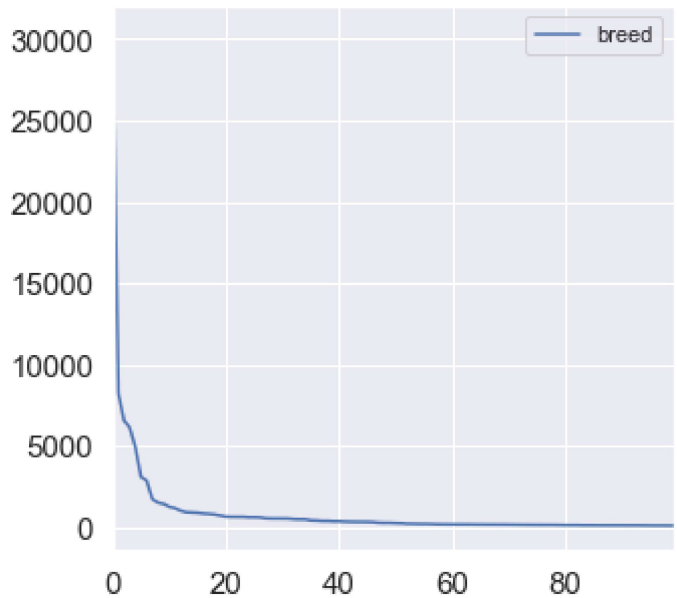


As expected, it looks like most of these "Other" animals are euthanized. It's important to consider that when we look at the total numbers of animals being euthanized, bats and other non-domesticated animals will be included in this number. It's also interesting that a small portion of these "Other" animals are adopted!

Next, let's take a closer look at the Breed and Color columns. They both have a wide variety of possible values, so let's identify which values have the highest frequency.

In [25]:
```
breeds = pd.DataFrame(clean_data['breed'].value_counts()).reset_index()
plt.figure(figsize=(20,10))
breeds.iloc[:100].plot(figsize=(5,5), fontsize=15)
```

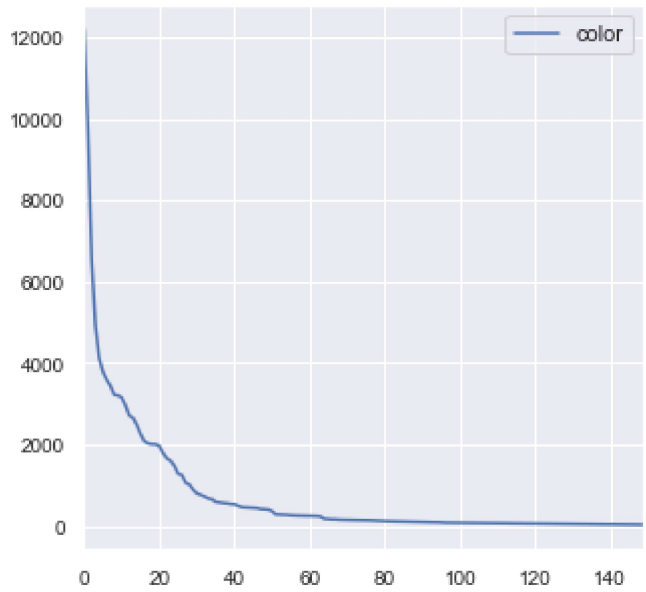
Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a228f97d0>

<Figure size 1440x720 with 0 Axes>
```



Based on this graph, we can see that the frequency of breeds tapers off drastically after the top 10 breeds. The imbalance means that we could probably make breeds after the top 20 in the "Other" category.

In [26]:
```
colors = pd.DataFrame(clean_data['color'].value_counts()).reset_index()
plt.figure(figsize=(30,30))
colors.iloc[:150].plot(figsize=(5,5), fontsize=10)
top_colors = colors.iloc[:50]
```

```
<Figure size 2160x2160 with 0 Axes>
```



Based on this graph, we can see that the frequency of colors tapers off drastically after the top 50 breeds. The imbalance means that we could probably make breeds after the top 60 in the "Other" category.

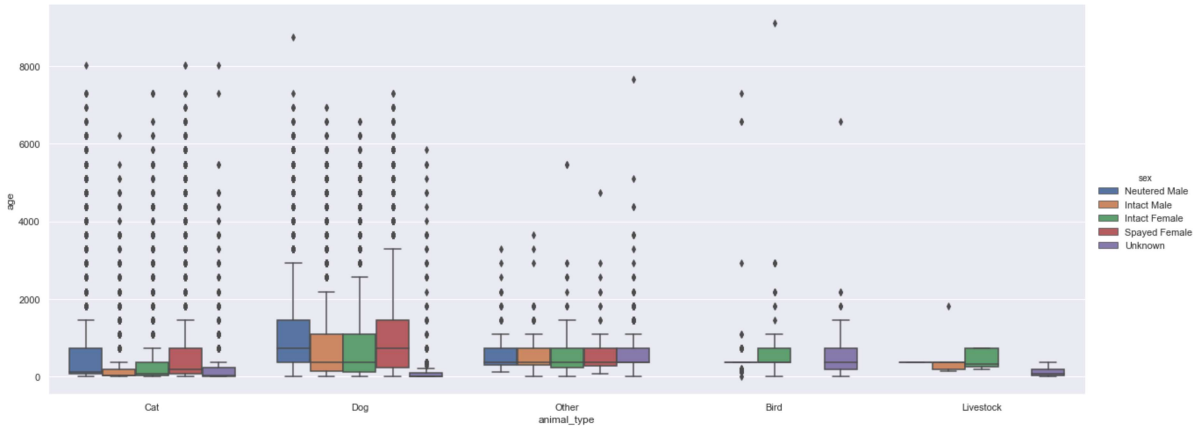
We will also examine the Age column and look at how old each animal type tends to be.

In [27]:
```
print('Mean age in weeks grouped by animal type')
x = clean_data.groupby(['animal_type'])['age'].mean()
print(x / 7)
print('')

print('Median age in weeks grouped by animal type')
y = clean_data.groupby(['animal_type'])['age'].median()
print(y / 7)
print('')

print('Standard Deviation of age in weeks grouped by animal type')
z = clean_data.groupby(['animal_type'])['age'].std()
print(z / 7)
print('')
```

```
Mean age in weeks grouped by animal type
animal_type
Bird          71.784257
Cat           73.451573
Dog          137.798478
Livestock     58.773109
Other         66.377693
Name: age, dtype: float64

Median age in weeks grouped by animal type
animal_type
Bird          52.142857
Cat           12.857143
Dog          104.285714
Livestock     47.142857
Other         52.142857
Name: age, dtype: float64

Standard Deviation of age in weeks grouped by animal type
animal_type
Bird         110.497589
Cat          138.835555
Dog          157.667069
Livestock     61.142733
Other         56.261074
Name: age, dtype: float64
```

In [28]:
```
sns.catplot(x="animal_type", y="age",
                hue="sex",
                  data=clean_data, kind="box", height = 7, aspect = 2.5)
```

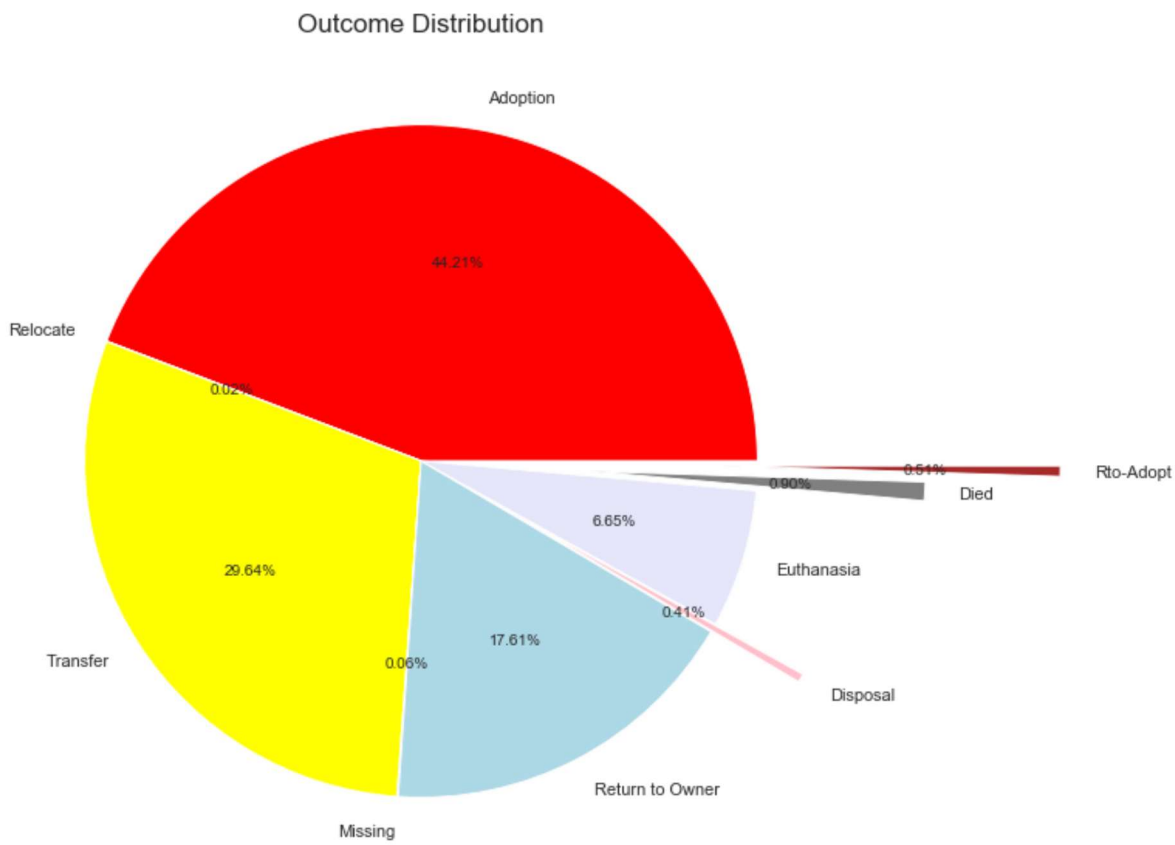
Out[28]: <seaborn.axisgrid.FacetGrid at 0x1a1f583b50>



Most animals in the shelter are relatively young (between one and two years old). The average age of dogs is higher, being over two years old. Dogs also have the most variation in age, followed by cats.

Next, let's look at the distribution of outcomes (class labels) across all records.

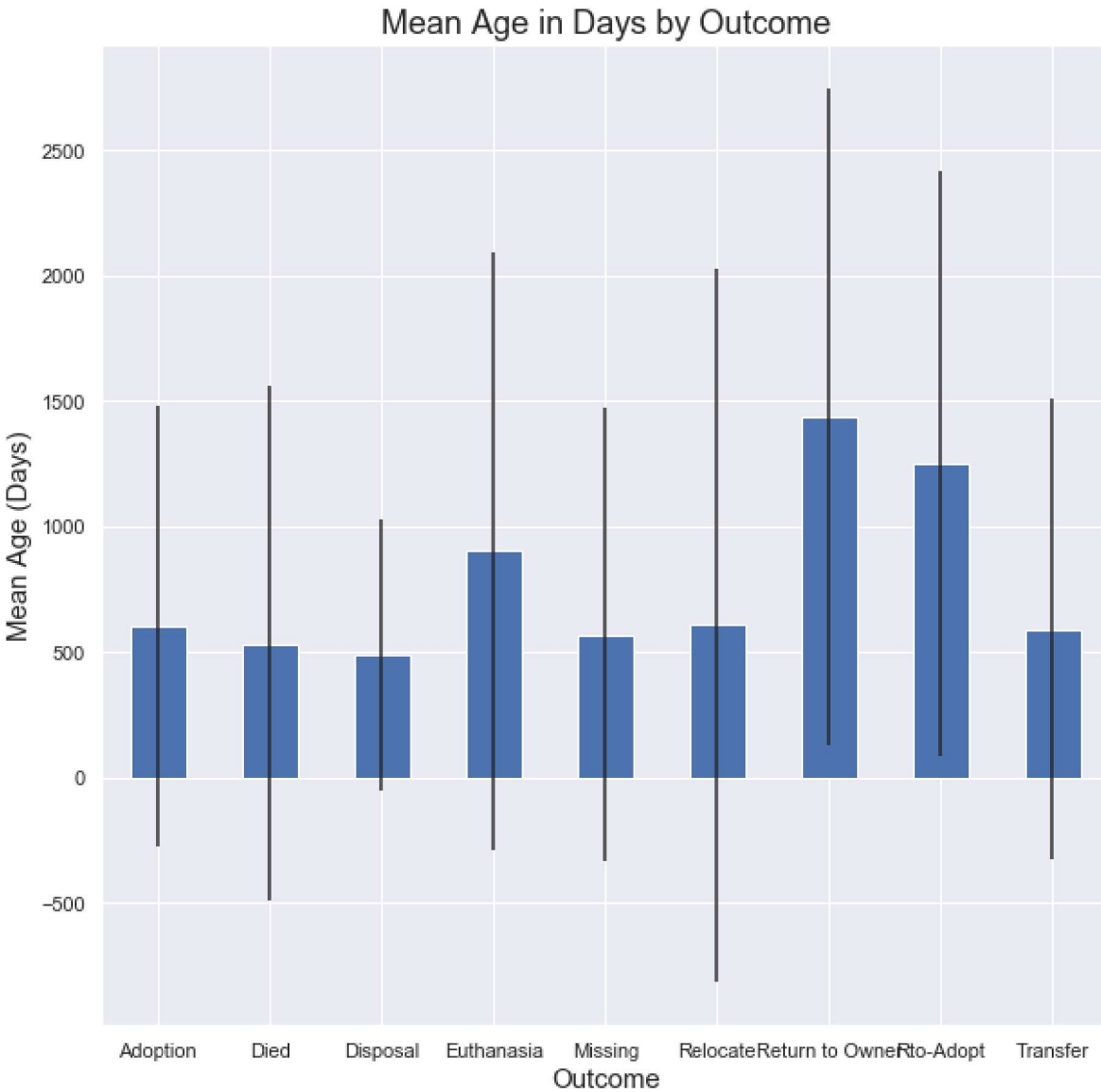
In [29]:
```
labels = ['Adoption', 'Relocate',  'Transfer', 'Missing', 'Return to Owner',
'Disposal', 'Euthanasia', 'Died', 'Rto-Adopt']
sizes = [51399, 20, 34460, 67, 20474, 482, 7728, 1049, 592]
colors = ['red','orange','yellow','green', 'lightblue', 'pink', 'lavender', 'g
ray', 'brown']
explode = [0, 0, 0, 0, 0, 0.3, 0, 0.5, 0.9]
fig1, ax1 = plt.subplots(figsize=(10,10))
ax1.pie(sizes, labels=labels, colors=colors,explode=explode, autopct='%1.2f%%'
, labeldistance=1.1)
plt.title('Outcome Distribution', fontsize=17)
plt.show()
```



The pie chart shows us that most animals at the shelter are adopted, transferred, or returned to owner. It's good to know that only a small portion die in the shelter or have to be put to sleep.

Next, let's look at the ages of animals per outcome.

In [30]:
```
meanie = clean_data.groupby(['outcome'])['age'].mean()
standy = clean_data.groupby(['outcome'])['age'].std()

meanie.plot.bar(x="animal_type",y='age', yerr=standy, rot=0, figsize=(10,10),
fontsize=11, color='b')

plt.xlabel('Outcome', fontsize=15)
plt.ylabel('Mean Age (Days)', fontsize=15)
plt.title('Mean Age in Days by Outcome', fontsize=18)
plt.show()
```



It appears that animals that are returned to their owner are older- probably because they are pets and have grown up with a family. Younger animals, on the other hand, are more prone to dying or being adopted/transferred.

FEATURE ENGINEERING:

Overview: We merged categories in the class label to avoid redundancy of labels, for a total of 4 classes: Adoption, Transfer, Return to Owner, and Death. We created an "Other" attribute for the Breed and Color columns for attributes that had a low frequency. This would simplify things for data analysis since breeds/colors that have very low frequency would not be necessarily helpful, and might just serve as noise. We also took out the dob column because it's redundant with the age column.

In [31]:
```
clean_data['outcome'].value_counts()
```

Out[31]:
```
Adoption           51399
Transfer           34460
Return to Owner    20474
Euthanasia          7728
Died                1049
Rto-Adopt            592
Disposal             482
Missing               67
Relocate              20
Name: outcome, dtype: int64
```

In [32]:
```
#we can merge some of these outcome types that mean the same thing:
#relocate --> transfer
#disposal --> died
#rto-adopt --> adoption

#take out missing outcomes
#4 final categories: adoption, transfer, return to owner, died.
def merge(outcome):
    if ((outcome == 'Euthanasia') or (outcome == 'Disposal')):
        return 'Died'
    elif (outcome == 'Relocate'):
        return 'Transfer'
    elif (outcome == 'Rto-Adopt'):
        return 'Adoption'
    elif (outcome == 'Missing'):
        return None
    else:
        return outcome
```

In [33]:
```
clean_data['class'] = clean_data.apply(lambda x: merge(x['outcome']), axis = 1
)
```

In [34]:
```
clean_data.head()
```

Out[34]:

	id	name	date	dob	outcome	animal_type	sex	age	
0	A794011	Chunk	2019-05-08 18:20:00	05/02/2017	Rto-Adopt	Cat	Neutered Male	730.0	Domestic
1	A776359	Gizmo	2018-07-18 16:02:00	07/12/2017	Adoption	Dog	Neutered Male	365.0	C Sho
2	A720371	Moose	2016-02-13 17:59:00	10/08/2015	Adoption	Dog	Neutered Male	120.0	Shepherd/
3	A674754	None	2014-03-18 11:47:00	03/12/2014	Transfer	Cat	Intact Male	6.0	Domestic
4	A689724	*Donatello	2014-10-18 18:52:00	08/01/2014	Adoption	Cat	Neutered Male	60.0	Domestic

In [35]:
```
clean_data['class'].value_counts()
```

Out[35]:
```
Adoption           51991
Transfer           34480
Return to Owner    20474
Died                9259
Name: class, dtype: int64
```

In [36]:
```
#drop missing outcomes and drop outcome column
print(clean_data['class'].isna().sum())
clean_data = clean_data.dropna(subset = ['class'])
clean_data = clean_data.drop(columns = ['outcome'])
clean_data.shape
```

```
67
```

Out[36]:
```
(116204, 10)
```

We also wanted to separate the date into month, day, year, day of the week, and time, so that we can get as much value out of this column as possible.

In [37]:
```
import datetime as dt
#separate date column into month, day, year, day of the week, and time.
clean_data['day'] = clean_data.date.dt.day
clean_data['month'] = clean_data.date.dt.month
clean_data['year'] = clean_data.date.dt.year
clean_data['time'] = clean_data.date.dt.time
clean_data['weekday'] = clean_data.date.dt.weekday
```

In [38]:
```
clean_data.head()
```

Out[38]:

	id	name	date	dob	animal_type	sex	age	breed
0	A794011	Chunk	2019-05-08 18:20:00	05/02/2017	Cat	Neutered Male	730.0	Domestic Shorthair Mix
1	A776359	Gizmo	2018-07-18 16:02:00	07/12/2017	Dog	Neutered Male	365.0	Chihuahua Shorthair Mix
2	A720371	Moose	2016-02-13 17:59:00	10/08/2015	Dog	Neutered Male	120.0	Anatol Shepherd/Labrador Retriever
3	A674754	None	2014-03-18 11:47:00	03/12/2014	Cat	Intact Male	6.0	Domestic Shorthair Mix
4	A689724	*Donatello	2014-10-18 18:52:00	08/01/2014	Cat	Neutered Male	60.0	Domestic Shorthair Mix

Another problem we ran into was that there are many different breeds and colors as possible attributes. A lot of these breeds and colors have a very low frequency (i.e. they only show up in the data once). We categorized any breeds or colors under thresholds as "Other".

In [39]:
```
#threshold for breeds, if under threshold --> "Other". threshold here is top 3
0!
top_breeds = breeds.iloc[:50]
top_breeds.columns = ['topbreed', 'count']
breed_updated_data = pd.merge(clean_data, top_breeds, how = 'left', left_on =
'breed', right_on = 'topbreed')
```

In [40]:
```
print(breed_updated_data.topbreed.isna().sum())
breed_updated_data.topbreed = breed_updated_data.topbreed.fillna(value = 'Othe
r')
```

25826

In [41]:
```
breed_updated_data.head()
```

Out[41]:

	id	name	date	dob	animal_type	sex	age	breed
0	A794011	Chunk	2019-05-08 18:20:00	05/02/2017	Cat	Neutered Male	730.0	Domestic Shorthair Mix
1	A776359	Gizmo	2018-07-18 16:02:00	07/12/2017	Dog	Neutered Male	365.0	Chihuahua Shorthair Mix
2	A720371	Moose	2016-02-13 17:59:00	10/08/2015	Dog	Neutered Male	120.0	Anatol Shepherd/Labrador Retriever
3	A674754	None	2014-03-18 11:47:00	03/12/2014	Cat	Intact Male	6.0	Domestic Shorthair Mix
4	A689724	*Donatello	2014-10-18 18:52:00	08/01/2014	Cat	Neutered Male	60.0	Domestic Shorthair Mix

In [42]:
```
#drop regular breed column
breed_updated_data = breed_updated_data.drop(columns = ['breed'])

#threshold for colors, if under threshold --> "Other". threshold here is top 5
0!
top_colors.columns = ['topcolor', 'count']
color_updated_data = pd.merge(breed_updated_data, top_colors, how = 'left', le
ft_on = 'color', right_on = 'topcolor')
```

In [43]:
```
print(color_updated_data.topcolor.isna().sum())
color_updated_data.topcolor = color_updated_data.topcolor.fillna(value = 'Othe
r')
```

12294

In [44]:
```
color_updated_data.head()
```

Out[44]:

	id	name	date	dob	animal_type	sex	age	color	cla
0	A794011	Chunk	2019-05-08 18:20:00	05/02/2017	Cat	Neutered Male	730.0	Brown Tabby/White	Adopti
1	A776359	Gizmo	2018-07-18 16:02:00	07/12/2017	Dog	Neutered Male	365.0	White/Brown	Adopti
2	A720371	Moose	2016-02-13 17:59:00	10/08/2015	Dog	Neutered Male	120.0	Buff	Adopti
3	A674754	None	2014-03-18 11:47:00	03/12/2014	Cat	Intact Male	6.0	Orange Tabby	Trans
4	A689724	*Donatello	2014-10-18 18:52:00	08/01/2014	Cat	Neutered Male	60.0	Black	Adopti

In [45]:
```
#dataset is ready for model!
final_data = color_updated_data.filter(items = ['animal_type', 'sex', 'age',
'topbreed',
                                                'topcolor', 'day', 'month', 'y
ear', 'time','class'])
final_data.columns = ['animal_type', 'sex', 'age', 'breed', 'color', 'day', 'm
onth', 'year', 'time','class']
final_data.head()
```

Out[45]:

	animal_type	sex	age	breed	color	day	month	year	time	clas
0	Cat	Neutered Male	730.0	Domestic Shorthair Mix	Brown Tabby/White	8	5	2019	18:20:00	Adoptio
1	Dog	Neutered Male	365.0	Chihuahua Shorthair Mix	White/Brown	18	7	2018	16:02:00	Adoptio
2	Dog	Neutered Male	120.0	Other	Buff	13	2	2016	17:59:00	Adoptio
3	Cat	Intact Male	6.0	Domestic Shorthair Mix	Orange Tabby	18	3	2014	11:47:00	Transfe
4	Cat	Neutered Male	60.0	Domestic Shorthair Mix	Black	18	10	2014	18:52:00	Adoptio

Finally, we also grouped the ages to be either young or adult, to make it easier to classify.

In [67]:
```
#We'll group by ages first:
#Re-classifying the age data by young versus adult, with young being a year ol
d or younger,
#and adults being older than a year. This makes it easier to do comparisons wi
th the data.
grouped_age_data = final_data.copy()

def age_range(age):
    if age <= 365:
        return 'Young'
    elif age > 365:
        return 'Adult'
    else:
        return age

grouped_age_data['age_range'] = grouped_age_data.apply(lambda x: age_range(x[
'age']), axis = 1)
grouped_age_data = grouped_age_data.drop(columns = 'age')
```

In [68]:
```
grouped_age_data['age_range'].value_counts()
```

Out[68]:
```
Young    70476
Adult    45728
Name: age_range, dtype: int64
```

After feature engineering, we went back and did a bit more data exploration with our new features.

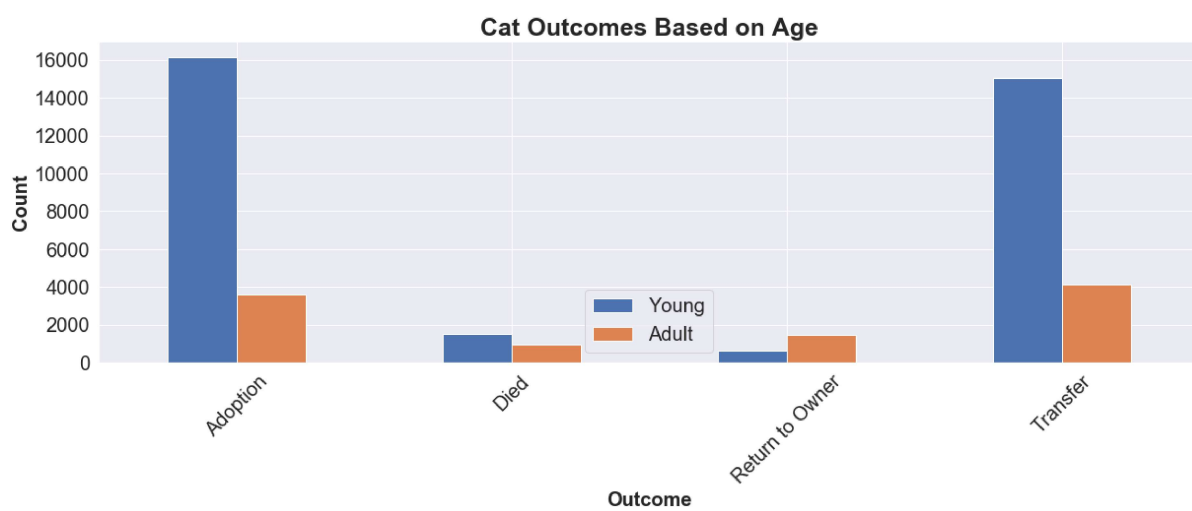
Data Exploration Part 2

First, let's analyze only the adopted animals. Since the largest animal type groups are dogs and cats (and they tend to be adopted), we'll focus on each of those animal types separately.

In [69]:
```
cats_data = grouped_age_data[ grouped_age_data['animal_type'] == 'Cat' ]
```

In [64]:
```
count_class = cats_data.groupby(["class", "age_range"]).size().reset_index(nam
e="count")
classes = count_class['class'].unique()

adult_count = count_class[count_class.age_range == 'Adult']
young_count = count_class[count_class.age_range == 'Young']

young = young_count['count'].to_list()
adult = adult_count['count'].to_list()
index = classes
df = pd.DataFrame({'Young': young,
                   'Adult': adult}, index=index)
ax = df.plot.bar(figsize=(20,6), rot=45, fontsize=20)
ax.set_xlabel('Outcome', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Cat Outcomes Based on Age', fontsize=25, fontweight='bold')
ax.legend(prop={'size': 20})
print(count_class)
```

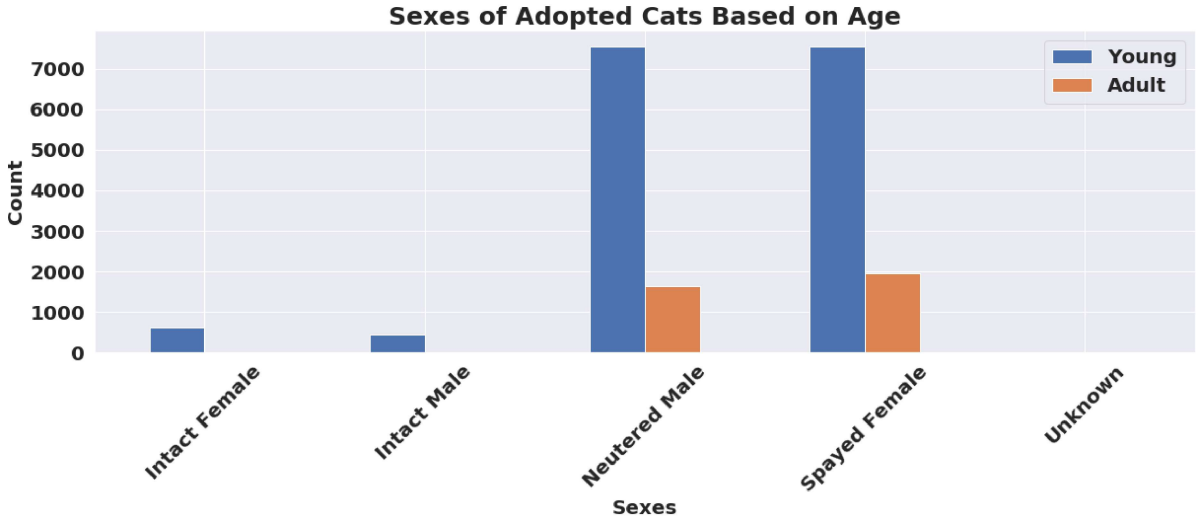
```
             class age_range  count
0         Adoption     Adult   3612
1         Adoption     Young  16163
2             Died     Adult    971
3             Died     Young   1527
4  Return to Owner     Adult   1450
5  Return to Owner     Young    633
6         Transfer     Adult   4137
7         Transfer     Young  15034
```



Most cats are being adopted or being transferred, with the majority of adopted and transferred cats being young. Cat adoptions are 81.73% young cats.

In [319]:
```
## dataset that only contains the adopted cats
cats_adopted_data = cats_data[ cats_data['class'] == 'Adoption' ]

count_age = cats_adopted_data.groupby(["sex", "age_range"]).size().reset_index
(name="count")
sexes = count_age.sex.unique()

adult_count = count_age[count_age.age_range == 'Adult']
young_count = count_age[count_age.age_range == 'Young']

young = young_count['count'].to_list()
adult = adult_count['count'].to_list()
index = sexes
df = pd.DataFrame({'Young': young,
                   'Adult': adult}, index=index)
ax = df.plot.bar(figsize=(20,6), rot=45, fontsize=20)
ax.set_xlabel('Sexes', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Sexes of Adopted Cats Based on Age', fontsize=25, fontweight='bo
ld')
ax.legend(prop={'size': 20})
print(count_age)
```

```
             sex age_range  count
0  Intact Female     Adult     17
1  Intact Female     Young    612
2    Intact Male     Adult      8
3    Intact Male     Young    446
4  Neutered Male     Adult   1638
5  Neutered Male     Young   7548
6  Spayed Female     Adult   1948
7  Spayed Female     Young   7556
8        Unknown     Adult      1
9        Unknown     Young      1
```

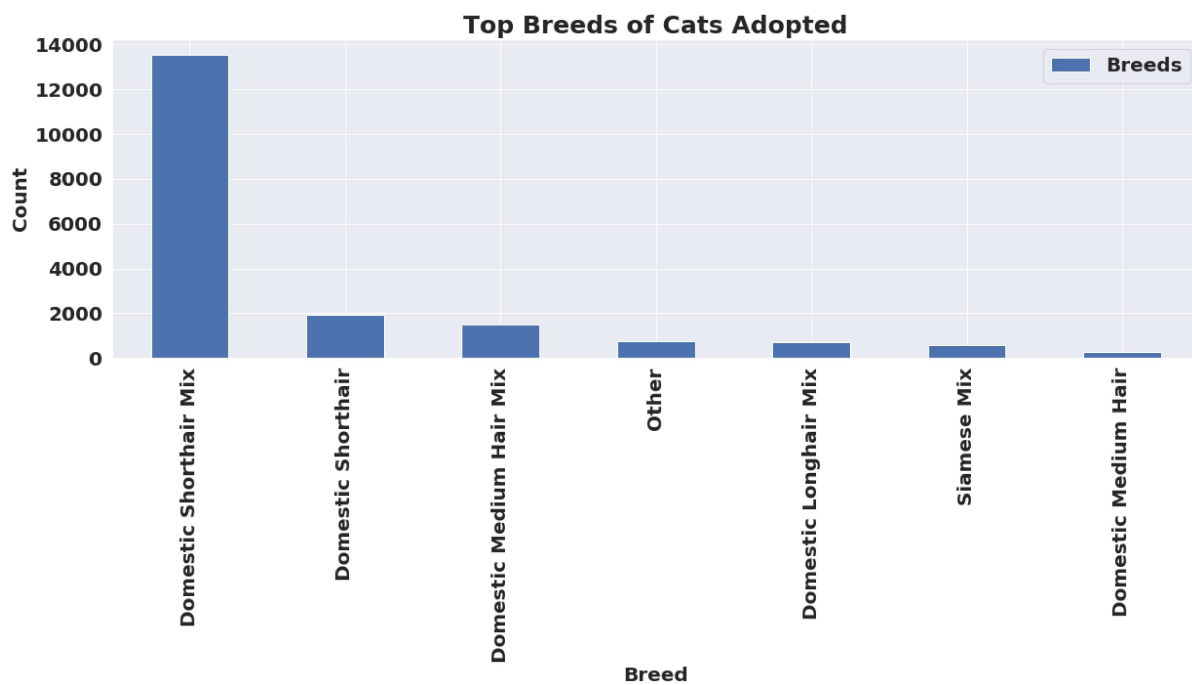


People are primarily adopting young spayed and neutered cats

In [384]:
```
top_cat_breeds = cats_adopted_data['breed'].value_counts().reset_index()

print(top_cat_breeds)

breeds = top_cat_breeds['breed'].to_list()
index = top_cat_breeds['index'].to_list()

df = pd.DataFrame({'Breeds': breeds}, index=index)
ax = df.plot.bar(figsize=(20,6), rot=90, fontsize=20, legend=False)
ax.set_xlabel('Breed', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Top Breeds of Cats Adopted', fontsize=25, fontweight='bold')
ax.legend(prop={'size': 20})
```

```
                      index  breed
0    Domestic Shorthair Mix  13545
1        Domestic Shorthair   1913
2  Domestic Medium Hair Mix   1476
3                     Other    764
4     Domestic Longhair Mix    731
5               Siamese Mix    592
6      Domestic Medium Hair    295
```

Out[384]:
```
<matplotlib.legend.Legend at 0x1a34760550>
```

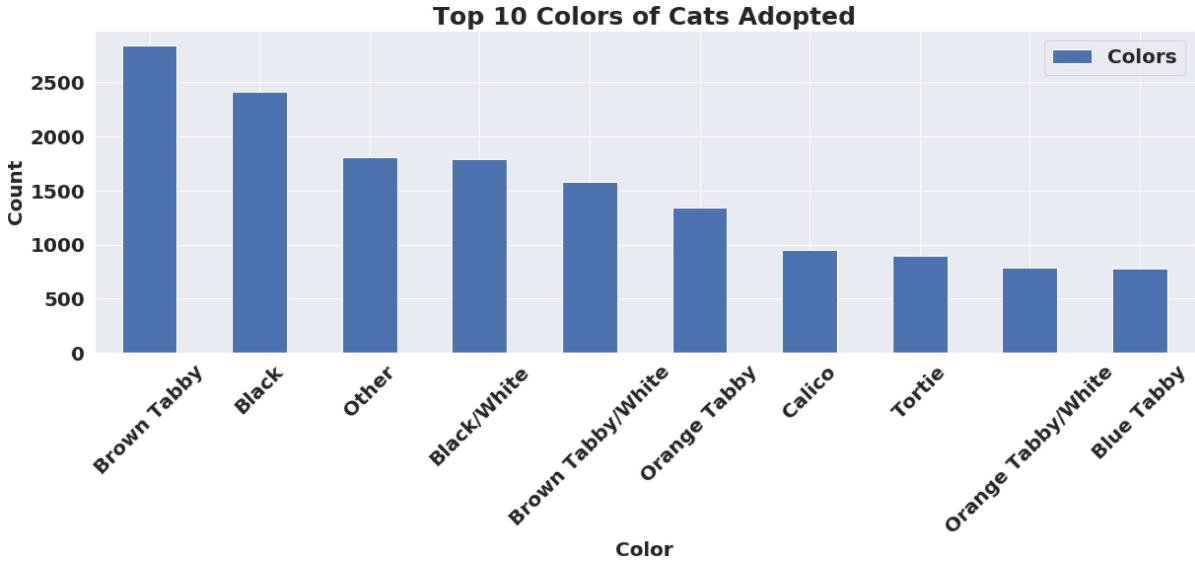


There was only 7 types of cat breeds that were adopted, with the Domestic Shorthair Mix being most common

In [385]:
```
cat_colors = cats_adopted_data['color'].value_counts().reset_index()
top_cat_colors = cat_colors[:10]
print(top_cat_colors)

colors = top_cat_colors['color'].to_list()
index = top_cat_colors['index'].to_list()

df = pd.DataFrame({'Colors': colors}, index=index)
ax = df.plot.bar(figsize=(20,6), rot=45, fontsize=20, legend=False)
ax.set_xlabel('Color', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Top 10 Colors of Cats Adopted', fontsize=25, fontweight='bold')
ax.legend(prop={'size': 20})
```

```
                index  color
0         Brown Tabby   2838
1               Black   2415
2               Other   1812
3         Black/White   1788
4   Brown Tabby/White   1583
5        Orange Tabby   1346
6              Calico    952
7              Tortie    894
8  Orange Tabby/White    783
9          Blue Tabby    774
```

Out[385]:
```
<matplotlib.legend.Legend at 0x1a5ab8b510>
```



Top 10 colors of cats adopted has a good variety

Dataset that only contains dogs

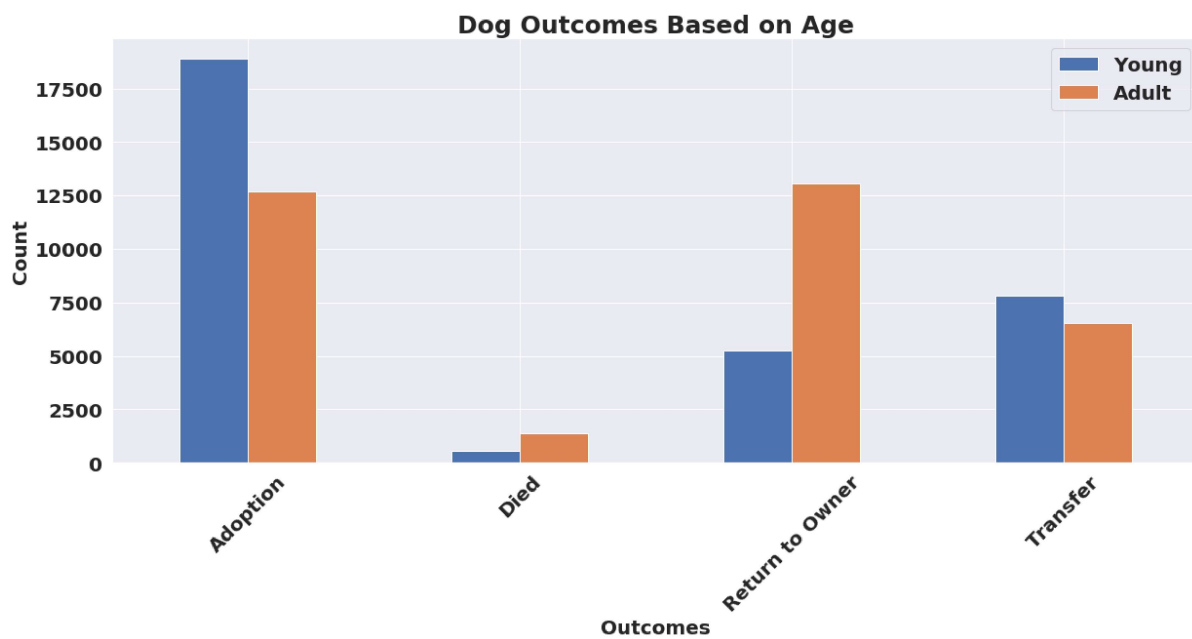
In [158]:
```
dogs_data = grouped_age_data[ grouped_age_data['animal_type'] == 'Dog' ]
dogs_data.head()
```

Out[158]:

	animal_type	sex	breed	color	day	month	year	time	class	age
1	Dog	Neutered Male	Chihuahua Shorthair Mix	White/Brown	18	7	2018	16:02:00	Adoption	
2	Dog	Neutered Male	Other	Buff	13	2	2016	17:59:00	Adoption	
14	Dog	Neutered Male	Chihuahua Shorthair Mix	Tricolor	8	6	2018	13:04:00	Adoption	
15	Dog	Intact Male	Yorkshire Terrier Mix	Brown/Black	26	10	2017	15:22:00	Transfer	
16	Dog	Spayed Female	Other	White/Tan	25	2	2018	17:19:00	Return to Owner	

In [326]:
```
count_class = dogs_data.groupby(["class", "age_range"]).size().reset_index(nam
e="count")
classes = count_class['class'].unique()

adult_count = count_class[count_class.age_range == 'Adult']
young_count = count_class[count_class.age_range == 'Young']

young = young_count['count'].to_list()
adult = adult_count['count'].to_list()
index = classes
df = pd.DataFrame({'Young': young,
                   'Adult': adult}, index=index)
ax = df.plot.bar(figsize=(20,8), rot=45, fontsize=20)
ax.set_xlabel('Outcomes', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Dog Outcomes Based on Age', fontsize=25, fontweight='bold')
ax.legend(prop={'size': 20})
print(count_class)
```

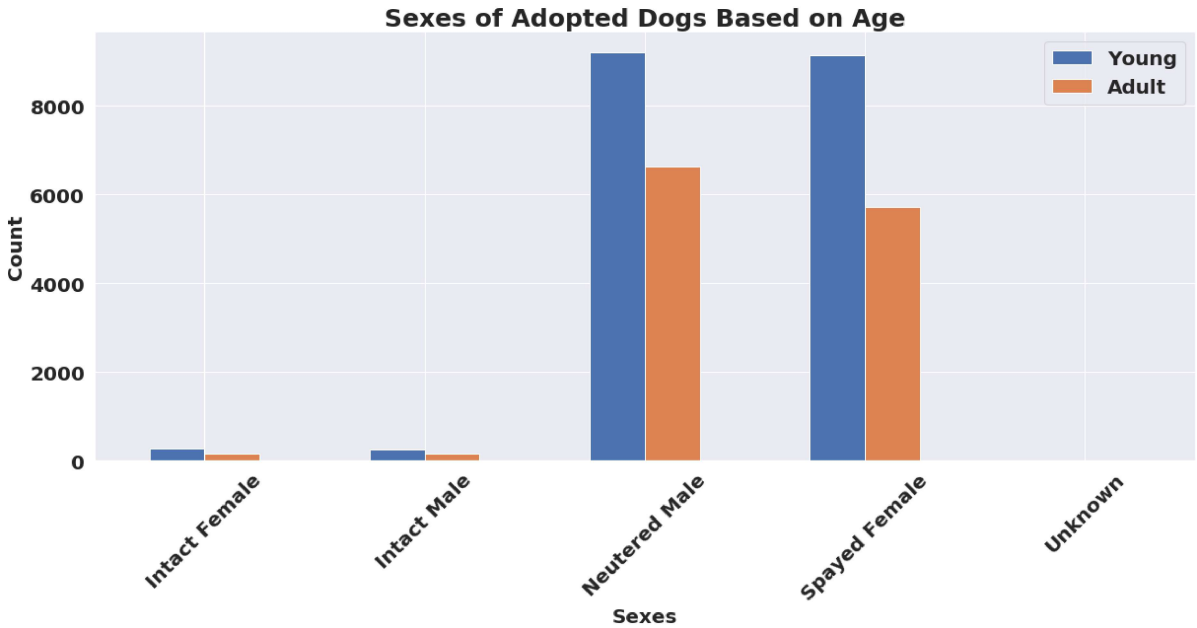
```
             class age_range  count
0         Adoption     Adult  12676
1         Adoption     Young  18889
2             Died     Adult   1399
3             Died     Young    575
4  Return to Owner     Adult  13066
5  Return to Owner     Young   5242
6         Transfer     Adult   6517
7         Transfer     Young   7799
```



Most dogs are being adopted or returned to owner, with the majority of adopted dogs being young, and the majority of dogs returned to owner are adults. Compared to cats, dogs have a smaller ratio of 59.84% of adopted dogs being young, compared to cat adoptions being 81.73% young.

In [336]:
```
## Dataset that only contains the adopted dogs
dogs_adopted_data = dogs_data[ dogs_data['class'] == 'Adoption' ]

count_age = dogs_adopted_data.groupby(["sex", "age_range"]).size().reset_index
(name="count")
sexes = count_age.sex.unique()

adult_count = count_age[count_age.age_range == 'Adult']
young_count = count_age[count_age.age_range == 'Young']

young = young_count['count'].to_list()
adult = adult_count['count'].to_list()
index = sexes
df = pd.DataFrame({'Young': young,
                   'Adult': adult}, index=index)

ax = df.plot.bar(figsize=(20,8), rot=45, fontsize=20)
ax.set_xlabel('Sexes', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Sexes of Adopted Dogs Based on Age', fontsize=25, fontweight='bo
ld')
ax.legend(prop={'size': 20})
print(count_age)
```

```
             sex age_range  count
0  Intact Female     Adult    163
1  Intact Female     Young    274
2    Intact Male     Adult    163
3    Intact Male     Young    259
4  Neutered Male     Adult   6634
5  Neutered Male     Young   9207
6  Spayed Female     Adult   5715
7  Spayed Female     Young   9148
8        Unknown     Adult      1
9        Unknown     Young      1
```

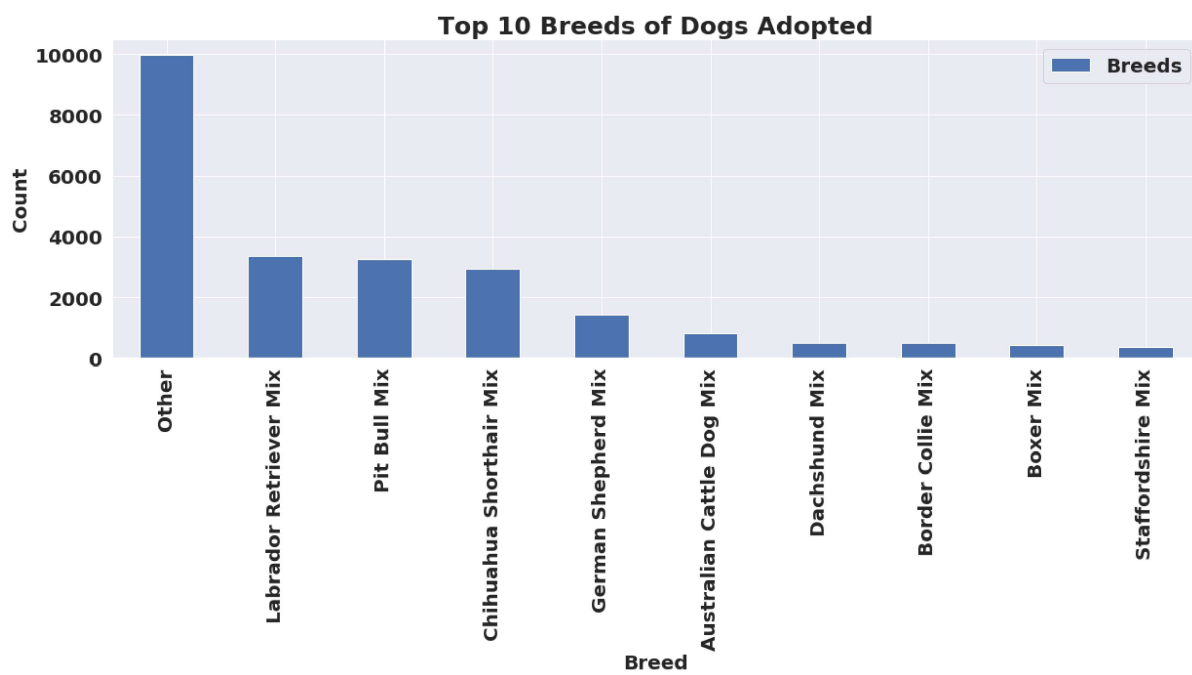


People are primarily adopting young spayed and neutered dogs

In [383]:
```
dog_breeds = dogs_adopted_data['breed'].value_counts().reset_index()
top_dog_breeds = dog_breeds[:10]
print(top_dog_breeds)

breeds = top_dog_breeds['breed'].to_list()
index = top_dog_breeds['index'].to_list()

df = pd.DataFrame({'Breeds': breeds}, index=index)
ax = df.plot.bar(figsize=(20,6), rot=90, fontsize=20, legend=False)
ax.set_xlabel('Breed', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Top 10 Breeds of Dogs Adopted', fontsize=25, fontweight='bold')
ax.legend(prop={'size': 20})
```

```
                      index  breed
0                     Other   9980
1     Labrador Retriever Mix   3354
2               Pit Bull Mix   3263
3    Chihuahua Shorthair Mix   2950
4        German Shepherd Mix   1441
5  Australian Cattle Dog Mix    809
6              Dachshund Mix    507
7          Border Collie Mix    489
8                  Boxer Mix    440
9          Staffordshire Mix    359
```

Out[383]:
```
<matplotlib.legend.Legend at 0x1a312b4150>
```

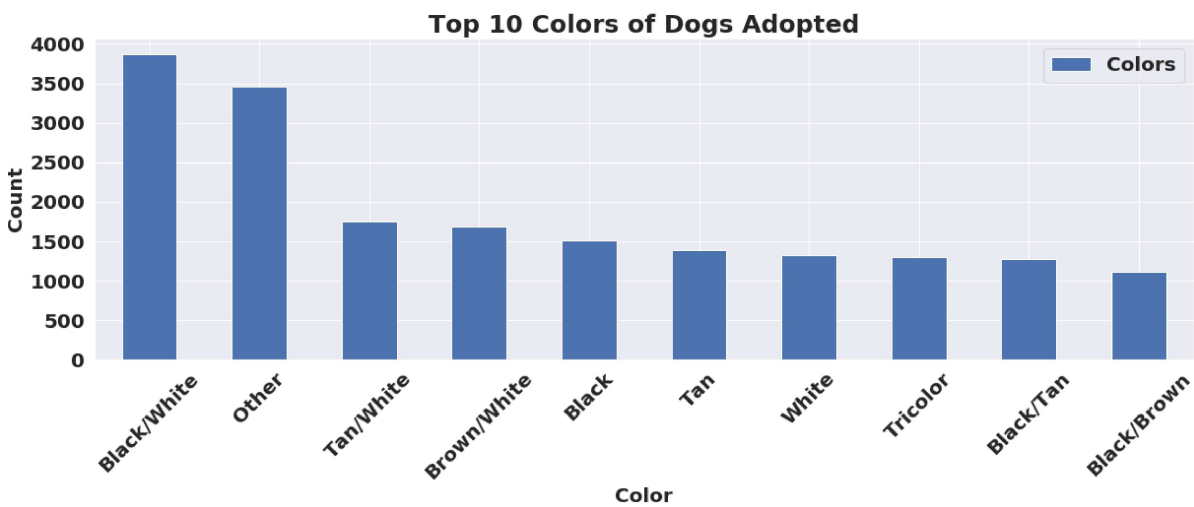


The top 10 breeds of adopted dogs. Other is what some people refer to as "mixed", or sometimes a derogatory term such as "mutt" is used. The top three that are not Other are Labrador Retriever Mix, Pit Bull Mix, and Chihuahua Shorthair Mix

In [379]:
```
dog_colors = dogs_adopted_data['color'].value_counts().reset_index()
top_dog_colors = dog_colors[:10]
print(top_dog_colors)

colors = top_dog_colors['color'].to_list()
index = top_dog_colors['index'].to_list()

df = pd.DataFrame({'Colors': colors}, index=index)
ax = df.plot.bar(figsize=(20,6), rot=45, fontsize=20, legend=False)
ax.set_xlabel('Color', fontsize=20, fontweight='bold')
ax.set_ylabel('Count', fontsize=20, fontweight='bold')
ax.set_title('Top 10 Colors of Dogs Adopted', fontsize=25, fontweight='bold')
ax.legend(prop={'size': 20})
```

```
          index  color
0  Black/White   3873
1        Other   3462
2    Tan/White   1748
3  Brown/White   1686
4        Black   1512
5          Tan   1383
6        White   1328
7     Tricolor   1296
8    Black/Tan   1271
9  Black/Brown   1114
```

Out[379]:
```
<matplotlib.legend.Legend at 0x1a365f4590>
```

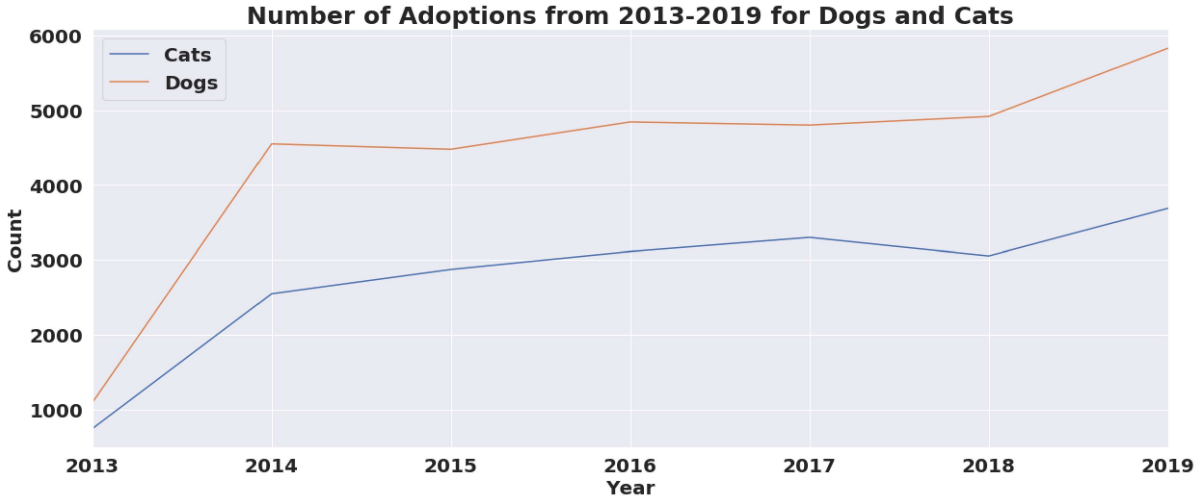


Top 10 colors of adopted dogs

In [361]:
```
# removed 2020 data due to incomplete year results, which altered the comparis
on with years
cats_adopted_data = cats_adopted_data[cats_adopted_data['year'] != 2020]
dogs_adopted_data = dogs_adopted_data[dogs_adopted_data['year'] != 2020]
```

In [363]:
```
cats_adopted_years = cats_adopted_data['year'].value_counts().reset_index()
dogs_adopted_years = dogs_adopted_data['year'].value_counts().reset_index()

cats_adopted_years = cats_adopted_years.sort_values(by='index')
dogs_adopted_years = dogs_adopted_years.sort_values(by='index')


df = pd.DataFrame({
   'Cats': cats_adopted_years['year'].to_list(),
    'Dogs':dogs_adopted_years['year'].to_list()},
    cats_adopted_years['index'].to_list())
lines = df.plot.line(figsize=(20,8), fontsize=20)
lines.set_xlabel('Year', fontsize=20, fontweight='bold')
lines.set_ylabel('Count', fontsize=20, fontweight='bold')
lines.set_title('Number of Adoptions from 2013-2019 for Dogs and Cats', fontsi
ze=25, fontweight='bold')
lines.legend(prop={'size': 20})
```

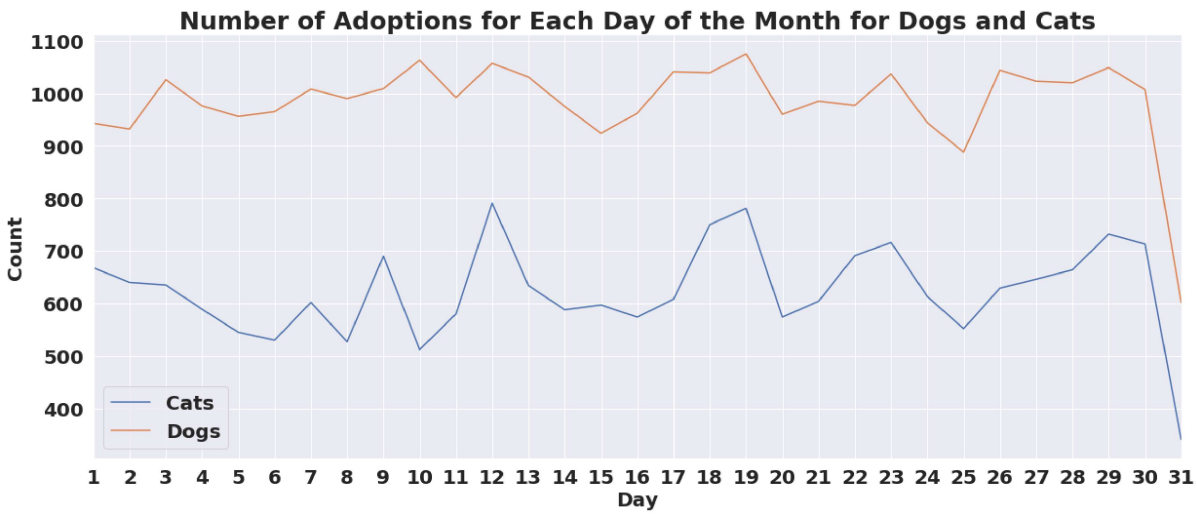
Out[363]:
```
<matplotlib.legend.Legend at 0x1a308a9650>
```



There are more dogs than cats that are adopted every year, however the adoption rates are similar and are increasing every year, except in 2018

In [364]:
```
cats_adopted_days = cats_adopted_data['day'].value_counts().reset_index()
dogs_adopted_days = dogs_adopted_data['day'].value_counts().reset_index()

cats_adopted_days = cats_adopted_days.sort_values(by='index')
dogs_adopted_days = dogs_adopted_days.sort_values(by='index')


df = pd.DataFrame({
   'Cats': cats_adopted_days['day'].to_list(),
    'Dogs':dogs_adopted_days['day'].to_list()},
    cats_adopted_days['index'].to_list())
lines = df.plot.line(figsize=(20,8), fontsize=20, xticks=cats_adopted_days['in
dex'])
lines.set_xlabel('Day', fontsize=20, fontweight='bold')
lines.set_ylabel('Count', fontsize=20, fontweight='bold')
lines.set_title('Number of Adoptions for Each Day of the Month for Dogs and Ca
ts', fontsize=25, fontweight='bold')
lines.legend(prop={'size': 20})
```

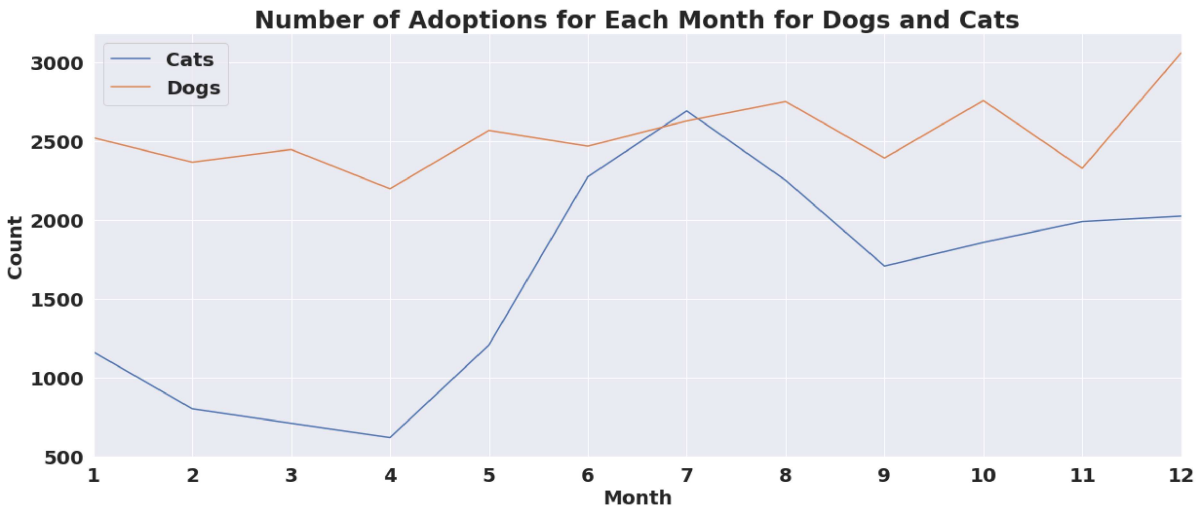
Out[364]: <matplotlib.legend.Legend at 0x1a31c5be10>



Cats and dogs are adopted at varying rates throughout the month, although there is some evidence that there are increased numbers on the weekends due to some spikes in the data being 7 days apart. Cats and dogs are also rarely adopted at the last day of the month.

In [365]:
```
cats_adopted_months = cats_adopted_data['month'].value_counts().reset_index()
dogs_adopted_months = dogs_adopted_data['month'].value_counts().reset_index()

cats_adopted_months = cats_adopted_months.sort_values(by='index')
dogs_adopted_months = dogs_adopted_months.sort_values(by='index')


df = pd.DataFrame({
   'Cats': cats_adopted_months['month'].to_list(),
    'Dogs':dogs_adopted_months['month'].to_list()},
    cats_adopted_months['index'].to_list())
lines = df.plot.line(figsize=(20,8), fontsize=20, xticks=cats_adopted_months[
'index'])
lines.set_xlabel('Month', fontsize=20, fontweight='bold')
lines.set_ylabel('Count', fontsize=20, fontweight='bold')
lines.set_title('Number of Adoptions for Each Month for Dogs and Cats', fontsi
ze=25, fontweight='bold')
lines.legend(prop={'size': 20})
```

Out[365]: <matplotlib.legend.Legend at 0x1a5ad2c490>



This line graph shows that for the most part dogs are adopted at a steady rate each year. Dog adoptions peak in December, which may be due to Christmas. Cats have a spike in adoption in July, as this is due to the increase in temperature, allowing them to have more litters.

source: *https://www.petful.com/animal-welfare/summer-is-near-and-so-is-kitten-season/ (https://www.petful.com/animal-welfare/summer-is-near-and-so-is-kitten-season/)*

Modeling:

Decision Tree Classifier

Hot Encode the categorical variables of data, drop time because of the datetime datatype. Also take the first 1000 adoption and first 1000 deaths to make a new dataset that deals with the class imbalance

In [70]:
```
encoded_data = final_data.copy()
```

In [71]:
```
grouped_age_data = grouped_age_data.drop(columns = ['day', 'month'])
```

In [171]:
```
grouped_adoption = grouped_age_data[ grouped_age_data['class'] == 'Adoption' ]
grouped_died = grouped_age_data[ grouped_age_data['class'] == 'Died' ]
grouped_adoption = grouped_adoption.head(1000)
grouped_died = grouped_died.head(1000)

balanced_data = pd.concat([grouped_adoption, grouped_died])
numbers = balanced_data['class'].value_counts().reset_index()
print(numbers)
balanced_data.head()
```

```
      index  class
0      Died   1000
1  Adoption   1000
```

Out[171]:

	animal_type	sex	breed	color	year	time	class	age_range
0	Cat	Neutered Male	Domestic Shorthair Mix	Brown Tabby/White	2019	18:20:00	Adoption	Adult
1	Dog	Neutered Male	Chihuahua Shorthair Mix	White/Brown	2018	16:02:00	Adoption	Young
2	Dog	Neutered Male	Other	Buff	2016	17:59:00	Adoption	Young
4	Cat	Neutered Male	Domestic Shorthair Mix	Black	2014	18:52:00	Adoption	Young
5	Cat	Neutered Male	Domestic Shorthair Mix	Other	2014	16:59:00	Adoption	Young

In [73]:
```
from sklearn.preprocessing import LabelEncoder

one_encoded_data = pd.get_dummies(balanced_data, columns= ["animal_type","sex"
,"breed","color", "age_range", "class"])
one_encoded_data = one_encoded_data.drop(columns = ['class_Died','time','year'
])

one_encoded_data.head()
```

Out[73]:

	animal_type_Bird	animal_type_Cat	animal_type_Dog	animal_type_Other	sex_Intact Female	sex_Intac Mal
0	0	1	0	0	0	(
1	0	0	1	0	0	(
2	0	0	1	0	0	(
4	0	1	0	0	0	(
5	0	1	0	0	0	(

5 rows × 114 columns

Separate data into features x and labels y

In [74]:
```
data_Y = one_encoded_data['class_Adoption']
data_X = one_encoded_data.drop(['class_Adoption'], axis = 1)
print(data_X.shape)
print(data_Y.shape)
```

```
(2000, 113)
(2000,)
```

split x and y into training and test sets

In [75]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(data_X, data_Y, test_size
=.2)
print("X_Train", X_train.shape, "X_Test", X_test.shape)
print("Y_Train", Y_train.shape, "Y_Test", Y_test.shape)
```

```
X_Train (1600, 113) X_Test (400, 113)
Y_Train (1600,) Y_Test (400,)
```

scale X data for PCA and perform PCA

In [76]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

print()
print(type(X_train_scaled))
print(X_train_scaled.shape)

# make sure the data got scaled correctly
print(X_train_scaled[0])
print("mean=", X_train_scaled.mean())
print('std=', X_train_scaled.std())
```

```
<class 'numpy.ndarray'>
(1600, 113)
[-0.11250879 -0.68500937  1.21680736 -0.60045334 -0.29738086 -0.29237251
  1.51399507 -0.58504861 -0.62554324 -0.05598925 -0.03537746 -0.06628895
 -0.06628895 -0.04334192 -0.10050378 -0.04334192 -0.05598925 -0.26770839
 -0.31448545 -0.06135239 -0.03537746 -0.07930516 -0.07930516 -0.06628895
 -0.06135239 -0.05006262 -0.07088812 -0.19220048 -0.02500782 -0.10666772
 -0.08692914 -0.05006262 -0.16618427 -0.18509419 -0.54074681 -0.06628895
 -0.14734777 -0.03537746 -0.04334192 -0.07521183 -0.19220048 -0.04334192
 -0.06628895 -0.05598925 -0.50390326 -0.08692914  4.24824894 -0.03537746
 -0.06628895 -0.07521183 -0.16618427 -0.17011439 -0.05006262 -0.06628895
 -0.02500782 -0.09395338 -0.02500782 -0.05598925 -0.06628895 -0.06135239
 -0.30356247 -0.1451184  -0.12598816 -0.12598816 -0.04334192 -0.31448545
 -0.10962532 -0.09728167 -0.08692914 -0.15597367  2.47206616 -0.07521183
 -0.12076694 -0.2309211  -0.14056247 -0.17586311 -0.03537746 -0.13586604
 -0.05006262 -0.12852402 -0.06135239 -0.08320212 -0.06628895 -0.07521183
 -0.04334192 -0.15171652 -0.14734777 -0.03537746 -0.07521183 -0.14285714
 -0.12598816 -0.32517073 -0.14056247 -0.10962532 -0.06628895 -0.05006262
 -0.15806323 -0.08320212 -0.1382327  -0.09728167 -0.12076694 -0.12852402
 -0.15385846 -0.1334605  -0.06135239 -0.12076694 -0.06135239 -0.06628895
 -0.08320212 -0.10666772 -0.05598925 -0.76430257  0.76430257]
mean= -1.1711379162417581e-17
std= 1.0
```

In [77]:
```
import numpy as np
from sklearn.decomposition import PCA

# fit PCA on training data X
pca = PCA(n_components=0.95, svd_solver='full')
pca_train = pca.fit_transform(X_train_scaled)

#transform the test set to the principal component space that was created
pca_test = pca.transform(X_test_scaled)

print(pca_train.shape)
print(pca_test.shape)
pca_test
```

```
(1600, 95)
(400, 95)
```

Out[77]:
```
array([[-1.27374286,  3.27899258, -0.06923806, ...,  0.18879113,
        -0.27084066, -0.38043761],
       [-1.13954855, -2.20295258, -1.36488205, ...,  0.72430594,
        -0.99265309,  0.73701818],
       [-1.328063  , -2.62572918, -0.72192514, ...,  0.30098823,
        -1.0062695 ,  0.63050415],
       ...,
       [-0.99710568,  2.70239863, -0.16176421, ...,  0.34268439,
        -0.0519852 , -0.22418766],
       [ 2.56298032, -0.85637285, -1.53531266, ...,  0.68010326,
        -0.39841562,  0.13172153],
       [-1.45887716,  3.08641563,  0.32294445, ..., -0.01306118,
        -0.37206944,  0.38061663]])
```

In [78]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

clf = DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(pca_train, Y_train)
tree.plot_tree(clf)
```

Out[78]:
```
[Text(251.390625, 211.4, 'X[0] <= -0.192\nentropy = 1.0\nsamples = 1600\nvalu
e = [790, 810]'),
 Text(182.59553571428572, 199.32, 'X[2] <= 0.122\nentropy = 0.861\nsamples =
1111\nvalue = [316, 795]'),
 Text(114.09107142857142, 187.24, 'X[25] <= 0.071\nentropy = 0.997\nsamples =
550\nvalue = [258, 292]'),
 Text(56.79642857142857, 175.16, 'X[74] <= -0.009\nentropy = 0.905\nsamples =
284\nvalue = [91, 193]'),
 Text(29.228571428571428, 163.07999999999998, 'X[10] <= -0.213\nentropy = 1.0
\nsamples = 119\nvalue = [60, 59]'),
 Text(13.285714285714285, 151.0, 'X[41] <= -0.283\nentropy = 0.906\nsamples =
56\nvalue = [18, 38]'),
 Text(7.9714285714285715, 138.92000000000002, 'X[24] <= -0.774\nentropy = 0.8
9\nsamples = 13\nvalue = [9, 4]'),
 Text(5.314285714285714, 126.84, 'X[29] <= -0.042\nentropy = 0.985\nsamples =
7\nvalue = [3, 4]'),
 Text(2.657142857142857, 114.75999999999999, 'entropy = 0.0\nsamples = 2\nval
ue = [0, 2]'),
 Text(7.9714285714285715, 114.75999999999999, 'entropy = 0.971\nsamples = 5\n
value = [3, 2]'),
 Text(10.628571428571428, 126.84, 'entropy = 0.0\nsamples = 6\nvalue = [6,
0]'),
 Text(18.599999999999998, 138.92000000000002, 'X[41] <= 1.896\nentropy = 0.74
\nsamples = 43\nvalue = [9, 34]'),
 Text(15.942857142857143, 126.84, 'X[2] <= -1.365\nentropy = 0.61\nsamples =
40\nvalue = [6, 34]'),
 Text(13.285714285714285, 114.75999999999999, 'entropy = 0.0\nsamples = 12\nv
alue = [0, 12]'),
 Text(18.599999999999998, 114.75999999999999, 'X[94] <= -0.393\nentropy = 0.7
5\nsamples = 28\nvalue = [6, 22]'),
 Text(13.285714285714285, 102.67999999999999, 'X[64] <= -1.169\nentropy = 0.9
18\nsamples = 6\nvalue = [4, 2]'),
 Text(10.628571428571428, 90.6, 'entropy = 1.0\nsamples = 4\nvalue = [2,
2]'),
 Text(15.942857142857143, 90.6, 'entropy = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(23.914285714285715, 102.67999999999999, 'X[42] <= 0.831\nentropy = 0.43
9\nsamples = 22\nvalue = [2, 20]'),
 Text(21.257142857142856, 90.6, 'entropy = 0.0\nsamples = 9\nvalue = [0,
9]'),
 Text(26.57142857142857, 90.6, 'X[14] <= -0.076\nentropy = 0.619\nsamples = 1
3\nvalue = [2, 11]'),
 Text(23.914285714285715, 78.52000000000001, 'entropy = 0.469\nsamples = 10\n
value = [1, 9]'),
 Text(29.228571428571428, 78.52000000000001, 'entropy = 0.918\nsamples = 3\nv
alue = [1, 2]'),
 Text(21.257142857142856, 126.84, 'entropy = 0.0\nsamples = 3\nvalue = [3,
0]'),
 Text(45.17142857142857, 151.0, 'X[70] <= 0.823\nentropy = 0.918\nsamples = 6
3\nvalue = [42, 21]'),
 Text(42.51428571428571, 138.92000000000002, 'X[92] <= 0.841\nentropy = 0.985
\nsamples = 49\nvalue = [28, 21]'),
 Text(39.857142857142854, 126.84, 'X[35] <= 1.409\nentropy = 0.996\nsamples =
39\nvalue = [18, 21]'),
 Text(37.199999999999996, 114.75999999999999, 'X[63] <= 0.796\nentropy = 0.99
8\nsamples = 34\nvalue = [18, 16]'),
 Text(34.542857142857144, 102.67999999999999, 'X[34] <= -0.65\nentropy = 0.95
```

```
8\nsamples = 29\nvalue = [18, 11]'),
 Text(31.885714285714286, 90.6, 'entropy = 0.0\nsamples = 6\nvalue = [6,
0]'),
 Text(37.199999999999996, 90.6, 'X[30] <= -0.849\nentropy = 0.999\nsamples =
23\nvalue = [12, 11]'),
 Text(34.542857142857144, 78.52000000000001, 'entropy = 0.0\nsamples = 4\nval
ue = [0, 4]'),
 Text(39.857142857142854, 78.52000000000001, 'X[77] <= -0.217\nentropy = 0.94
9\nsamples = 19\nvalue = [12, 7]'),
 Text(34.542857142857144, 66.44, 'X[19] <= 0.424\nentropy = 0.994\nsamples =
11\nvalue = [5, 6]'),
 Text(31.885714285714286, 54.359999999999985, 'X[15] <= -0.19\nentropy = 0.97
1\nsamples = 10\nvalue = [4, 6]'),
 Text(29.228571428571428, 42.28, 'entropy = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(34.542857142857144, 42.28, 'X[35] <= 0.708\nentropy = 0.991\nsamples =
9\nvalue = [4, 5]'),
 Text(31.885714285714286, 30.19999999999999, 'X[44] <= 0.554\nentropy = 1.0\n
samples = 6\nvalue = [3, 3]'),
 Text(29.228571428571428, 18.120000000000005, 'entropy = 1.0\nsamples = 2\nva
lue = [1, 1]'),
 Text(34.542857142857144, 18.120000000000005, 'X[8] <= 1.728\nentropy = 1.0\n
samples = 4\nvalue = [2, 2]'),
 Text(31.885714285714286, 6.039999999999992, 'entropy = 1.0\nsamples = 2\nval
ue = [1, 1]'),
 Text(37.199999999999996, 6.039999999999992, 'entropy = 1.0\nsamples = 2\nval
ue = [1, 1]'),
 Text(37.199999999999996, 30.19999999999999, 'entropy = 0.918\nsamples = 3\nv
alue = [1, 2]'),
 Text(37.199999999999996, 54.359999999999985, 'entropy = 0.0\nsamples = 1\nva
lue = [1, 0]'),
 Text(45.17142857142857, 66.44, 'X[41] <= -1.745\nentropy = 0.544\nsamples =
8\nvalue = [7, 1]'),
 Text(42.51428571428571, 54.359999999999985, 'entropy = 0.918\nsamples = 3\nv
alue = [2, 1]'),
 Text(47.82857142857143, 54.359999999999985, 'entropy = 0.0\nsamples = 5\nval
ue = [5, 0]'),
 Text(39.857142857142854, 102.67999999999999, 'entropy = 0.0\nsamples = 5\nva
lue = [0, 5]'),
 Text(42.51428571428571, 114.75999999999999, 'entropy = 0.0\nsamples = 5\nval
ue = [0, 5]'),
 Text(45.17142857142857, 126.84, 'entropy = 0.0\nsamples = 10\nvalue = [10,
0]'),
 Text(47.82857142857143, 138.92000000000002, 'entropy = 0.0\nsamples = 14\nva
lue = [14, 0]'),
 Text(84.36428571428571, 163.07999999999998, 'X[2] <= -1.55\nentropy = 0.697
\nsamples = 165\nvalue = [31, 134]'),
 Text(70.41428571428571, 151.0, 'X[21] <= 0.814\nentropy = 0.962\nsamples = 5
7\nvalue = [22, 35]'),
 Text(58.457142857142856, 138.92000000000002, 'X[75] <= -1.024\nentropy = 0.9
89\nsamples = 32\nvalue = [18, 14]'),
 Text(55.8, 126.84, 'entropy = 0.0\nsamples = 6\nvalue = [6, 0]'),
 Text(61.114285714285714, 126.84, 'X[12] <= 0.313\nentropy = 0.996\nsamples =
26\nvalue = [12, 14]'),
 Text(55.8, 114.75999999999999, 'X[55] <= 0.154\nentropy = 0.964\nsamples = 1
8\nvalue = [11, 7]'),
 Text(53.14285714285714, 102.67999999999999, 'X[54] <= -0.019\nentropy = 0.97
```

```
7\nsamples = 17\nvalue = [10, 7]'),
 Text(47.82857142857143, 90.6, 'X[80] <= -0.349\nentropy = 1.0\nsamples = 8\n
value = [4, 4]'),
 Text(45.17142857142857, 78.52000000000001, 'entropy = 1.0\nsamples = 6\nvalu
e = [3, 3]'),
 Text(50.48571428571428, 78.52000000000001, 'entropy = 1.0\nsamples = 2\nvalu
e = [1, 1]'),
 Text(58.457142857142856, 90.6, 'X[72] <= -0.589\nentropy = 0.918\nsamples =
9\nvalue = [6, 3]'),
 Text(55.8, 78.52000000000001, 'entropy = 0.918\nsamples = 3\nvalue = [2,
1]'),
 Text(61.114285714285714, 78.52000000000001, 'X[53] <= -0.019\nentropy = 0.91
8\nsamples = 6\nvalue = [4, 2]'),
 Text(58.457142857142856, 66.44, 'entropy = 0.918\nsamples = 3\nvalue = [2,
1]'),
 Text(63.77142857142857, 66.44, 'entropy = 0.918\nsamples = 3\nvalue = [2,
1]'),
 Text(58.457142857142856, 102.67999999999999, 'entropy = 0.0\nsamples = 1\nva
lue = [1, 0]'),
 Text(66.42857142857143, 114.75999999999999, 'X[49] <= 0.109\nentropy = 0.544
\nsamples = 8\nvalue = [1, 7]'),
 Text(63.77142857142857, 102.67999999999999, 'entropy = 0.918\nsamples = 3\nv
alue = [1, 2]'),
 Text(69.08571428571429, 102.67999999999999, 'entropy = 0.0\nsamples = 5\nval
ue = [0, 5]'),
 Text(82.37142857142857, 138.92000000000002, 'X[13] <= -1.247\nentropy = 0.63
4\nsamples = 25\nvalue = [4, 21]'),
 Text(79.71428571428571, 126.84, 'X[81] <= 0.229\nentropy = 0.918\nsamples =
12\nvalue = [4, 8]'),
 Text(77.05714285714285, 114.75999999999999, 'X[39] <= 0.03\nentropy = 0.845
\nsamples = 11\nvalue = [3, 8]'),
 Text(74.39999999999999, 102.67999999999999, 'entropy = 0.722\nsamples = 5\nv
alue = [1, 4]'),
 Text(79.71428571428571, 102.67999999999999, 'entropy = 0.918\nsamples = 6\nv
alue = [2, 4]'),
 Text(82.37142857142857, 114.75999999999999, 'entropy = 0.0\nsamples = 1\nval
ue = [1, 0]'),
 Text(85.02857142857142, 126.84, 'entropy = 0.0\nsamples = 13\nvalue = [0, 1
3]'),
 Text(98.31428571428572, 151.0, 'X[52] <= 0.002\nentropy = 0.414\nsamples = 1
08\nvalue = [9, 99]'),
 Text(95.65714285714286, 138.92000000000002, 'X[55] <= 1.944\nentropy = 0.61
\nsamples = 60\nvalue = [9, 51]'),
 Text(93.0, 126.84, 'X[24] <= 0.583\nentropy = 0.531\nsamples = 58\nvalue =
[7, 51]'),
 Text(87.68571428571428, 114.75999999999999, 'X[19] <= -2.027\nentropy = 0.27
6\nsamples = 42\nvalue = [2, 40]'),
 Text(85.02857142857142, 102.67999999999999, 'X[47] <= 0.011\nentropy = 0.811
\nsamples = 8\nvalue = [2, 6]'),
 Text(82.37142857142857, 90.6, 'entropy = 0.811\nsamples = 4\nvalue = [1,
3]'),
 Text(87.68571428571428, 90.6, 'entropy = 0.811\nsamples = 4\nvalue = [1,
3]'),
 Text(90.34285714285714, 102.67999999999999, 'entropy = 0.0\nsamples = 34\nva
lue = [0, 34]'),
 Text(98.31428571428572, 114.75999999999999, 'X[70] <= -0.665\nentropy = 0.89
6\nsamples = 16\nvalue = [5, 11]'),
```

```
 Text(95.65714285714286, 102.67999999999999, 'entropy = 0.0\nsamples = 1\nval
ue = [1, 0]'),
 Text(100.97142857142856, 102.67999999999999, 'X[71] <= 0.198\nentropy = 0.83
7\nsamples = 15\nvalue = [4, 11]'),
 Text(98.31428571428572, 90.6, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(103.62857142857142, 90.6, 'X[62] <= 0.539\nentropy = 0.89\nsamples = 13
\nvalue = [4, 9]'),
 Text(100.97142857142856, 78.52000000000001, 'X[41] <= -0.006\nentropy = 0.84
5\nsamples = 11\nvalue = [3, 8]'),
 Text(98.31428571428572, 66.44, 'X[38] <= -1.101\nentropy = 0.811\nsamples =
8\nvalue = [2, 6]'),
 Text(95.65714285714286, 54.359999999999985, 'entropy = 0.811\nsamples = 4\nv
alue = [1, 3]'),
 Text(100.97142857142856, 54.359999999999985, 'entropy = 0.811\nsamples = 4\n
value = [1, 3]'),
 Text(103.62857142857142, 66.44, 'entropy = 0.918\nsamples = 3\nvalue = [1,
2]'),
 Text(106.28571428571428, 78.52000000000001, 'entropy = 1.0\nsamples = 2\nval
ue = [1, 1]'),
 Text(98.31428571428572, 126.84, 'entropy = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(100.97142857142856, 138.92000000000002, 'entropy = 0.0\nsamples = 48\nv
alue = [0, 48]'),
 Text(171.38571428571427, 175.16, 'X[24] <= -0.012\nentropy = 0.952\nsamples
= 266\nvalue = [167, 99]'),
 Text(144.15, 163.07999999999998, 'X[74] <= 1.212\nentropy = 0.986\nsamples =
130\nvalue = [56, 74]'),
 Text(141.49285714285713, 151.0, 'X[51] <= 0.028\nentropy = 1.0\nsamples = 11
4\nvalue = [56, 58]'),
 Text(123.55714285714285, 138.92000000000002, 'X[11] <= -0.363\nentropy = 0.9
14\nsamples = 73\nvalue = [24, 49]'),
 Text(111.6, 126.84, 'X[50] <= -0.025\nentropy = 0.976\nsamples = 22\nvalue =
[13, 9]'),
 Text(108.94285714285714, 114.75999999999999, 'entropy = 0.0\nsamples = 5\nva
lue = [5, 0]'),
 Text(114.25714285714285, 114.75999999999999, 'X[91] <= -0.296\nentropy = 0.9
98\nsamples = 17\nvalue = [8, 9]'),
 Text(111.6, 102.67999999999999, 'entropy = 0.0\nsamples = 3\nvalue = [0,
3]'),
 Text(116.91428571428571, 102.67999999999999, 'X[92] <= 0.667\nentropy = 0.98
5\nsamples = 14\nvalue = [8, 6]'),
 Text(114.25714285714285, 90.6, 'X[50] <= 0.255\nentropy = 1.0\nsamples = 12
\nvalue = [6, 6]'),
 Text(111.6, 78.52000000000001, 'X[87] <= -0.861\nentropy = 0.994\nsamples =
11\nvalue = [5, 6]'),
 Text(108.94285714285714, 66.44, 'entropy = 0.918\nsamples = 3\nvalue = [1,
2]'),
 Text(114.25714285714285, 66.44, 'X[41] <= 0.103\nentropy = 1.0\nsamples = 8
\nvalue = [4, 4]'),
 Text(111.6, 54.359999999999985, 'entropy = 1.0\nsamples = 4\nvalue = [2,
2]'),
 Text(116.91428571428571, 54.359999999999985, 'X[17] <= -0.23\nentropy = 1.0
\nsamples = 4\nvalue = [2, 2]'),
 Text(114.25714285714285, 42.28, 'entropy = 1.0\nsamples = 2\nvalue = [1,
1]'),
 Text(119.57142857142857, 42.28, 'entropy = 1.0\nsamples = 2\nvalue = [1,
1]'),
```

```
 Text(116.91428571428571, 78.52000000000001, 'entropy = 0.0\nsamples = 1\nval
ue = [1, 0]'),
 Text(119.57142857142857, 90.6, 'entropy = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(135.5142857142857, 126.84, 'X[80] <= -0.509\nentropy = 0.752\nsamples =
51\nvalue = [11, 40]'),
 Text(130.2, 114.75999999999999, 'X[68] <= 0.33\nentropy = 0.994\nsamples = 2
2\nvalue = [10, 12]'),
 Text(127.54285714285714, 102.67999999999999, 'X[75] <= -0.368\nentropy = 0.9
77\nsamples = 17\nvalue = [10, 7]'),
 Text(124.88571428571429, 90.6, 'entropy = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(130.2, 90.6, 'X[6] <= -0.712\nentropy = 0.996\nsamples = 13\nvalue =
[6, 7]'),
 Text(127.54285714285714, 78.52000000000001, 'entropy = 0.0\nsamples = 1\nval
ue = [1, 0]'),
 Text(132.85714285714286, 78.52000000000001, 'X[23] <= 0.114\nentropy = 0.98
\nsamples = 12\nvalue = [5, 7]'),
 Text(130.2, 66.44, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(135.5142857142857, 66.44, 'X[83] <= -0.22\nentropy = 0.994\nsamples = 1
1\nvalue = [5, 6]'),
 Text(132.85714285714286, 54.359999999999985, 'X[81] <= -0.285\nentropy = 1.0
\nsamples = 6\nvalue = [3, 3]'),
 Text(130.2, 42.28, 'entropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(135.5142857142857, 42.28, 'entropy = 1.0\nsamples = 4\nvalue = [2,
2]'),
 Text(138.17142857142858, 54.359999999999985, 'entropy = 0.971\nsamples = 5\n
value = [2, 3]'),
 Text(132.85714285714286, 102.67999999999999, 'entropy = 0.0\nsamples = 5\nva
lue = [0, 5]'),
 Text(140.82857142857142, 114.75999999999999, 'X[6] <= 1.02\nentropy = 0.216
\nsamples = 29\nvalue = [1, 28]'),
 Text(138.17142857142858, 102.67999999999999, 'entropy = 0.0\nsamples = 26\nv
alue = [0, 26]'),
 Text(143.48571428571427, 102.67999999999999, 'entropy = 0.918\nsamples = 3\n
value = [1, 2]'),
 Text(159.42857142857142, 138.92000000000002, 'X[74] <= -0.174\nentropy = 0.7
59\nsamples = 41\nvalue = [32, 9]'),
 Text(156.77142857142857, 126.84, 'X[89] <= 0.258\nentropy = 0.943\nsamples =
25\nvalue = [16, 9]'),
 Text(151.45714285714286, 114.75999999999999, 'X[42] <= 1.475\nentropy = 0.98
5\nsamples = 14\nvalue = [6, 8]'),
 Text(148.79999999999998, 102.67999999999999, 'X[2] <= -0.22\nentropy = 0.994
\nsamples = 11\nvalue = [6, 5]'),
 Text(146.14285714285714, 90.6, 'X[62] <= 0.35\nentropy = 0.918\nsamples = 9
\nvalue = [6, 3]'),
 Text(143.48571428571427, 78.52000000000001, 'entropy = 0.0\nsamples = 3\nval
ue = [3, 0]'),
 Text(148.79999999999998, 78.52000000000001, 'X[16] <= 0.018\nentropy = 1.0\n
samples = 6\nvalue = [3, 3]'),
 Text(146.14285714285714, 66.44, 'entropy = 1.0\nsamples = 2\nvalue = [1,
1]'),
 Text(151.45714285714286, 66.44, 'X[93] <= -0.419\nentropy = 1.0\nsamples = 4
\nvalue = [2, 2]'),
 Text(148.79999999999998, 54.359999999999985, 'entropy = 1.0\nsamples = 2\nva
lue = [1, 1]'),
 Text(154.1142857142857, 54.359999999999985, 'entropy = 1.0\nsamples = 2\nval
```

```
ue = [1, 1]'),
 Text(151.45714285714286, 90.6, 'entropy = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(154.1142857142857, 102.67999999999999, 'entropy = 0.0\nsamples = 3\nval
ue = [0, 3]'),
 Text(162.0857142857143, 114.75999999999999, 'X[90] <= -0.381\nentropy = 0.43
9\nsamples = 11\nvalue = [10, 1]'),
 Text(159.42857142857142, 102.67999999999999, 'entropy = 0.0\nsamples = 1\nva
lue = [0, 1]'),
 Text(164.74285714285713, 102.67999999999999, 'entropy = 0.0\nsamples = 10\nv
alue = [10, 0]'),
 Text(162.0857142857143, 126.84, 'entropy = 0.0\nsamples = 16\nvalue = [16,
0]'),
 Text(146.80714285714285, 151.0, 'entropy = 0.0\nsamples = 16\nvalue = [0, 1
6]'),
 Text(198.62142857142857, 163.07999999999998, 'X[23] <= 0.351\nentropy = 0.68
8\nsamples = 136\nvalue = [111, 25]'),
 Text(195.9642857142857, 151.0, 'X[92] <= 0.064\nentropy = 0.861\nsamples = 8
8\nvalue = [63, 25]'),
 Text(184.67142857142855, 138.92000000000002, 'X[86] <= 0.114\nentropy = 1.0
\nsamples = 41\nvalue = [21, 20]'),
 Text(178.0285714285714, 126.84, 'X[43] <= -0.301\nentropy = 0.906\nsamples =
28\nvalue = [9, 19]'),
 Text(172.7142857142857, 114.75999999999999, 'X[76] <= -0.091\nentropy = 0.96
1\nsamples = 13\nvalue = [8, 5]'),
 Text(170.05714285714285, 102.67999999999999, 'entropy = 0.811\nsamples = 4\n
value = [3, 1]'),
 Text(175.37142857142857, 102.67999999999999, 'X[68] <= -0.385\nentropy = 0.9
91\nsamples = 9\nvalue = [5, 4]'),
 Text(172.7142857142857, 90.6, 'entropy = 0.971\nsamples = 5\nvalue = [3,
2]'),
 Text(178.0285714285714, 90.6, 'X[57] <= -0.168\nentropy = 1.0\nsamples = 4\n
value = [2, 2]'),
 Text(175.37142857142857, 78.52000000000001, 'entropy = 1.0\nsamples = 2\nval
ue = [1, 1]'),
 Text(180.68571428571428, 78.52000000000001, 'entropy = 1.0\nsamples = 2\nval
ue = [1, 1]'),
 Text(183.34285714285713, 114.75999999999999, 'X[62] <= 0.592\nentropy = 0.35
3\nsamples = 15\nvalue = [1, 14]'),
 Text(180.68571428571428, 102.67999999999999, 'entropy = 0.0\nsamples = 11\nv
alue = [0, 11]'),
 Text(186.0, 102.67999999999999, 'entropy = 0.811\nsamples = 4\nvalue = [1,
3]'),
 Text(191.31428571428572, 126.84, 'X[92] <= -0.563\nentropy = 0.391\nsamples
= 13\nvalue = [12, 1]'),
 Text(188.65714285714284, 114.75999999999999, 'entropy = 0.0\nsamples = 1\nva
lue = [0, 1]'),
 Text(193.97142857142856, 114.75999999999999, 'entropy = 0.0\nsamples = 12\nv
alue = [12, 0]'),
 Text(207.25714285714284, 138.92000000000002, 'X[83] <= -0.144\nentropy = 0.4
89\nsamples = 47\nvalue = [42, 5]'),
 Text(201.94285714285712, 126.84, 'X[33] <= 0.126\nentropy = 1.0\nsamples = 8
\nvalue = [4, 4]'),
 Text(199.28571428571428, 114.75999999999999, 'X[30] <= 0.912\nentropy = 0.72
2\nsamples = 5\nvalue = [4, 1]'),
 Text(196.62857142857143, 102.67999999999999, 'entropy = 0.0\nsamples = 2\nva
lue = [2, 0]'),
```

```
 Text(201.94285714285712, 102.67999999999999, 'entropy = 0.918\nsamples = 3\n
value = [2, 1]'),
 Text(204.6, 114.75999999999999, 'entropy = 0.0\nsamples = 3\nvalue = [0,
3]'),
 Text(212.57142857142856, 126.84, 'X[43] <= -1.706\nentropy = 0.172\nsamples
= 39\nvalue = [38, 1]'),
 Text(209.9142857142857, 114.75999999999999, 'entropy = 0.65\nsamples = 6\nva
lue = [5, 1]'),
 Text(215.22857142857143, 114.75999999999999, 'entropy = 0.0\nsamples = 33\nv
alue = [33, 0]'),
 Text(201.2785714285714, 151.0, 'entropy = 0.0\nsamples = 48\nvalue = [48,
0]'),
 Text(251.1, 187.24, 'X[0] <= -1.313\nentropy = 0.48\nsamples = 561\nvalue =
[58, 503]'),
 Text(225.85714285714286, 175.16, 'X[88] <= -0.114\nentropy = 0.184\nsamples
= 286\nvalue = [8, 278]'),
 Text(223.2, 163.07999999999998, 'entropy = 0.0\nsamples = 121\nvalue = [0, 1
21]'),
 Text(228.5142857142857, 163.07999999999998, 'X[77] <= 0.227\nentropy = 0.28
\nsamples = 165\nvalue = [8, 157]'),
 Text(225.85714285714286, 151.0, 'X[81] <= -0.161\nentropy = 0.374\nsamples =
111\nvalue = [8, 103]'),
 Text(220.54285714285714, 138.92000000000002, 'X[35] <= -1.85\nentropy = 0.63
9\nsamples = 37\nvalue = [6, 31]'),
 Text(217.88571428571427, 126.84, 'entropy = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(223.2, 126.84, 'X[57] <= -0.011\nentropy = 0.513\nsamples = 35\nvalue =
[4, 31]'),
 Text(220.54285714285714, 114.75999999999999, 'entropy = 0.0\nsamples = 11\nv
alue = [0, 11]'),
 Text(225.85714285714286, 114.75999999999999, 'X[3] <= 0.217\nentropy = 0.65
\nsamples = 24\nvalue = [4, 20]'),
 Text(223.2, 102.67999999999999, 'entropy = 1.0\nsamples = 2\nvalue = [1,
1]'),
 Text(228.5142857142857, 102.67999999999999, 'X[49] <= -0.023\nentropy = 0.57
5\nsamples = 22\nvalue = [3, 19]'),
 Text(225.85714285714286, 90.6, 'X[84] <= -0.317\nentropy = 0.61\nsamples = 2
0\nvalue = [3, 17]'),
 Text(223.2, 78.52000000000001, 'entropy = 0.503\nsamples = 9\nvalue = [1,
8]'),
 Text(228.5142857142857, 78.52000000000001, 'X[70] <= -0.161\nentropy = 0.684
\nsamples = 11\nvalue = [2, 9]'),
 Text(225.85714285714286, 66.44, 'entropy = 0.65\nsamples = 6\nvalue = [1,
5]'),
 Text(231.17142857142855, 66.44, 'entropy = 0.722\nsamples = 5\nvalue = [1,
4]'),
 Text(231.17142857142855, 90.6, 'entropy = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(231.17142857142855, 138.92000000000002, 'X[1] <= 2.9\nentropy = 0.179\n
samples = 74\nvalue = [2, 72]'),
 Text(228.5142857142857, 126.84, 'entropy = 0.0\nsamples = 45\nvalue = [0, 4
5]'),
 Text(233.82857142857142, 126.84, 'X[26] <= -0.953\nentropy = 0.362\nsamples
= 29\nvalue = [2, 27]'),
 Text(231.17142857142855, 114.75999999999999, 'entropy = 0.469\nsamples = 10
\nvalue = [1, 9]'),
 Text(236.48571428571427, 114.75999999999999, 'X[39] <= 0.028\nentropy = 0.29
```

```
7\nsamples = 19\nvalue = [1, 18]'),
 Text(233.82857142857142, 102.67999999999999, 'entropy = 0.31\nsamples = 18\n
value = [1, 17]'),
 Text(239.14285714285714, 102.67999999999999, 'entropy = 0.0\nsamples = 1\nva
lue = [0, 1]'),
 Text(231.17142857142855, 151.0, 'entropy = 0.0\nsamples = 54\nvalue = [0, 5
4]'),
 Text(276.34285714285716, 175.16, 'X[14] <= 0.581\nentropy = 0.684\nsamples =
275\nvalue = [50, 225]'),
 Text(259.07142857142856, 163.07999999999998, 'X[12] <= 0.805\nentropy = 0.39
3\nsamples = 194\nvalue = [15, 179]'),
 Text(252.42857142857142, 151.0, 'X[93] <= 0.783\nentropy = 0.222\nsamples =
168\nvalue = [6, 162]'),
 Text(247.1142857142857, 138.92000000000002, 'X[16] <= -0.581\nentropy = 0.10
1\nsamples = 153\nvalue = [2, 151]'),
 Text(244.45714285714286, 126.84, 'X[26] <= 0.427\nentropy = 0.684\nsamples =
11\nvalue = [2, 9]'),
 Text(241.79999999999998, 114.75999999999999, 'entropy = 0.0\nsamples = 7\nva
lue = [0, 7]'),
 Text(247.1142857142857, 114.75999999999999, 'X[40] <= 0.504\nentropy = 1.0\n
samples = 4\nvalue = [2, 2]'),
 Text(244.45714285714286, 102.67999999999999, 'entropy = 0.918\nsamples = 3\n
value = [1, 2]'),
 Text(249.77142857142857, 102.67999999999999, 'entropy = 0.0\nsamples = 1\nva
lue = [1, 0]'),
 Text(249.77142857142857, 126.84, 'entropy = 0.0\nsamples = 142\nvalue = [0,
142]'),
 Text(257.74285714285713, 138.92000000000002, 'X[19] <= -0.096\nentropy = 0.8
37\nsamples = 15\nvalue = [4, 11]'),
 Text(255.0857142857143, 126.84, 'X[4] <= -0.392\nentropy = 0.985\nsamples =
7\nvalue = [4, 3]'),
 Text(252.42857142857142, 114.75999999999999, 'entropy = 0.971\nsamples = 5\n
value = [2, 3]'),
 Text(257.74285714285713, 114.75999999999999, 'entropy = 0.0\nsamples = 2\nva
lue = [2, 0]'),
 Text(260.4, 126.84, 'entropy = 0.0\nsamples = 8\nvalue = [0, 8]'),
 Text(265.7142857142857, 151.0, 'X[7] <= -0.625\nentropy = 0.931\nsamples = 2
6\nvalue = [9, 17]'),
 Text(263.0571428571428, 138.92000000000002, 'entropy = 0.0\nsamples = 5\nval
ue = [5, 0]'),
 Text(268.37142857142857, 138.92000000000002, 'X[8] <= -0.006\nentropy = 0.70
2\nsamples = 21\nvalue = [4, 17]'),
 Text(265.7142857142857, 126.84, 'X[94] <= -0.288\nentropy = 1.0\nsamples = 8
\nvalue = [4, 4]'),
 Text(263.0571428571428, 114.75999999999999, 'entropy = 0.0\nsamples = 2\nval
ue = [2, 0]'),
 Text(268.37142857142857, 114.75999999999999, 'X[47] <= -0.078\nentropy = 0.9
18\nsamples = 6\nvalue = [2, 4]'),
 Text(265.7142857142857, 102.67999999999999, 'entropy = 0.0\nsamples = 1\nval
ue = [0, 1]'),
 Text(271.0285714285714, 102.67999999999999, 'X[77] <= 0.685\nentropy = 0.971
\nsamples = 5\nvalue = [2, 3]'),
 Text(268.37142857142857, 90.6, 'entropy = 0.918\nsamples = 3\nvalue = [1,
2]'),
 Text(273.68571428571425, 90.6, 'entropy = 1.0\nsamples = 2\nvalue = [1,
1]'),
 Text(271.0285714285714, 126.84, 'entropy = 0.0\nsamples = 13\nvalue = [0, 1
```

```
3]'),
 Text(293.6142857142857, 163.07999999999998, 'X[76] <= -0.189\nentropy = 0.98
7\nsamples = 81\nvalue = [35, 46]'),
 Text(286.97142857142853, 151.0, 'X[11] <= 1.332\nentropy = 0.821\nsamples =
39\nvalue = [29, 10]'),
 Text(284.3142857142857, 138.92000000000002, 'X[34] <= -0.139\nentropy = 0.53
3\nsamples = 33\nvalue = [29, 4]'),
 Text(281.65714285714284, 126.84, 'entropy = 0.0\nsamples = 18\nvalue = [18,
0]'),
 Text(286.97142857142853, 126.84, 'X[27] <= 0.318\nentropy = 0.837\nsamples =
15\nvalue = [11, 4]'),
 Text(284.3142857142857, 114.75999999999999, 'X[74] <= 0.264\nentropy = 1.0\n
samples = 8\nvalue = [4, 4]'),
 Text(281.65714285714284, 102.67999999999999, 'X[13] <= -0.05\nentropy = 0.91
8\nsamples = 6\nvalue = [4, 2]'),
 Text(279.0, 90.6, 'entropy = 0.918\nsamples = 3\nvalue = [2, 1]'),
 Text(284.3142857142857, 90.6, 'entropy = 0.918\nsamples = 3\nvalue = [2,
1]'),
 Text(286.97142857142853, 102.67999999999999, 'entropy = 0.0\nsamples = 2\nva
lue = [0, 2]'),
 Text(289.62857142857143, 114.75999999999999, 'entropy = 0.0\nsamples = 7\nva
lue = [7, 0]'),
 Text(289.62857142857143, 138.92000000000002, 'entropy = 0.0\nsamples = 6\nva
lue = [0, 6]'),
 Text(300.25714285714287, 151.0, 'X[82] <= 0.265\nentropy = 0.592\nsamples =
42\nvalue = [6, 36]'),
 Text(294.9428571428571, 138.92000000000002, 'X[3] <= 0.757\nentropy = 0.303
\nsamples = 37\nvalue = [2, 35]'),
 Text(292.2857142857143, 126.84, 'entropy = 0.0\nsamples = 27\nvalue = [0, 2
7]'),
 Text(297.59999999999997, 126.84, 'entropy = 0.722\nsamples = 10\nvalue = [2,
8]'),
 Text(305.57142857142856, 138.92000000000002, 'X[59] <= -3.372\nentropy = 0.7
22\nsamples = 5\nvalue = [4, 1]'),
 Text(302.9142857142857, 126.84, 'entropy = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(308.2285714285714, 126.84, 'entropy = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(320.18571428571425, 199.32, 'X[2] <= 1.406\nentropy = 0.198\nsamples =
489\nvalue = [474, 15]'),
 Text(310.8857142857143, 187.24, 'X[4] <= 1.151\nentropy = 0.085\nsamples = 4
69\nvalue = [464, 5]'),
 Text(305.57142857142856, 175.16, 'X[94] <= 0.506\nentropy = 0.022\nsamples =
459\nvalue = [458, 1]'),
 Text(302.9142857142857, 163.07999999999998, 'entropy = 0.0\nsamples = 440\nv
alue = [440, 0]'),
 Text(308.2285714285714, 163.07999999999998, 'X[2] <= 1.164\nentropy = 0.297
\nsamples = 19\nvalue = [18, 1]'),
 Text(305.57142857142856, 151.0, 'entropy = 0.0\nsamples = 11\nvalue = [11,
0]'),
 Text(310.8857142857143, 151.0, 'entropy = 0.544\nsamples = 8\nvalue = [7,
1]'),
 Text(316.2, 175.16, 'X[94] <= -0.147\nentropy = 0.971\nsamples = 10\nvalue =
[6, 4]'),
 Text(313.54285714285714, 163.07999999999998, 'entropy = 0.0\nsamples = 6\nva
lue = [6, 0]'),
 Text(318.85714285714283, 163.07999999999998, 'entropy = 0.0\nsamples = 4\nva
```

```
lue = [0, 4]'),
 Text(329.48571428571427, 187.24, 'X[68] <= 0.138\nentropy = 1.0\nsamples = 2
0\nvalue = [10, 10]'),
 Text(326.8285714285714, 175.16, 'X[80] <= -0.612\nentropy = 0.439\nsamples =
11\nvalue = [1, 10]'),
 Text(324.1714285714286, 163.07999999999998, 'entropy = 0.918\nsamples = 3\nv
alue = [1, 2]'),
 Text(329.48571428571427, 163.07999999999998, 'entropy = 0.0\nsamples = 8\nva
lue = [0, 8]'),
 Text(332.1428571428571, 175.16, 'entropy = 0.0\nsamples = 9\nvalue = [9,
0]')]
```

print accuracy score of decision tree

In [80]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report


pred_Y = clf.predict(pca_test)
print(pred_Y)
accuracy_score(Y_test, pred_Y)
```

```
[0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1
 0 1 1 1 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1
 0 0 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1
 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 1 0 0 0 1 0 0 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1
 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 1 1
 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1
 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 0 1 1 1 1
 0 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 0
 0 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 0 0
 0 1 0 1 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1]
```

Out[80]: 0.8575

find more optimal parameters for decision tree

In [146]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score

clf1 = DecisionTreeClassifier(criterion='gini', min_samples_split=4, min_sampl
es_leaf=5)
clf1 = clf1.fit(pca_train, Y_train)
pred_Y1 = clf1.predict(pca_test)
print(pred_Y1)
x= accuracy_score(Y_test, pred_Y1)
print(x)

clf2 = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=5)
clf2 = clf2.fit(pca_train, Y_train)
pred_Y2 = clf2.predict(pca_test)
print(pred_Y2)
y= accuracy_score(Y_test, pred_Y2)
print(y)


clf3 = DecisionTreeClassifier(criterion='gini', max_depth=4)
clf3 = clf3.fit(pca_train, Y_train)
pred_Y3 = clf3.predict(pca_test)
print(pred_Y3)
z= accuracy_score(Y_test, pred_Y3)
print(z)

clf4 = DecisionTreeClassifier(criterion='gini', max_depth=4, min_samples_leaf=
5)
clf4 = clf4.fit(pca_train, Y_train)
pred_Y4 = clf4.predict(pca_test)
print(pred_Y4)
ab= accuracy_score(Y_test, pred_Y4)
print(ab)
```

```
[0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1
 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1
 0 0 1 1 1 0 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 1 0 1
 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0
 0 1 0 0 0 1 0 0 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1
 0 1 0 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 1 1
 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 0 1
 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1 0
 1 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 0 0
 0 1 1 0 1 0 1 1 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0
 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1]
0.865
[0 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1
 0 1 1 1 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1
 0 0 1 1 1 0 0 1 0 1 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1
 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0
 0 1 0 0 0 1 0 0 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1
 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 1 1
 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 0 1
 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1 1
 0 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 0
 0 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0
 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1]
0.8525
[0 1 1 1 1 0 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1
 0 1 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1
 0 0 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 1 1 1 0 1
 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0
 0 1 0 1 0 1 0 0 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1 0 0 1 1
 0 1 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1
 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 1 1 0 0 1 1 0 1
 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 1
 0 0 1 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0
 0 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0
 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0 1 1 0 0 0 0 1]
0.8625
[0 1 1 1 1 0 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1
 0 1 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1
 0 0 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 1 1 1 0 1
 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0
 0 1 0 1 0 1 0 0 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1 0 0 1 1
 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1
 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 1 1 0 0 1 1 0 1
 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 1
 0 0 1 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0
 0 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0
 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0 1 1 0 0 0 0 1]
0.86
```

The best parameters are using Gini, min_samples_split=4, min_samples_leaf=5

Naive Bayes:

implements the Gaussian Naive Bayes algorithm

In [148]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score

data_Y = one_encoded_data['class_Adoption']
data_X = one_encoded_data.drop(['class_Adoption'], axis = 1)
print(data_X.shape)
print(data_Y.shape)
data_X

clf = GaussianNB()

# Run a 10-fold cross validation
scores = cross_val_score(clf, data_X, data_Y, cv=10)

# display accuracy
scores.mean()
```

```
(2000, 113)
(2000,)
```

Out[148]:
```
0.5934999999999999
```

show the confusion matrix, precision, recall, and F1 score

In [149]:
```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


#Show the confusion matrix
y_pred = cross_val_predict(clf, data_X, data_Y, cv=10)
confusion_matrix1 = confusion_matrix(data_Y, y_pred)
print(confusion_matrix1)

#Classification Report
report = classification_report(data_Y, y_pred)
print(report)
```

```
[[950  50]
 [763 237]]
              precision    recall  f1-score   support

           0       0.55      0.95      0.70      1000
           1       0.83      0.24      0.37      1000

    accuracy                           0.59      2000
   macro avg       0.69      0.59      0.53      2000
weighted avg       0.69      0.59      0.53      2000
```

KNN Classifier:

normalize the dataset's features

In [150]:
```
from sklearn.preprocessing import MinMaxScaler

df_scaled = MinMaxScaler().fit_transform(data_X)

df_new = pd.DataFrame(data=df_scaled)
print(df_new.head())
```

```
   0    1    2    3    4    5    6    7    8    9    ...  103  104  105  106
\
0  0.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0
1  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  ...  0.0  0.0  1.0  0.0
2  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0
3  0.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0
4  0.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0

   107  108  109  110  111  112
0  0.0  0.0  0.0  0.0  1.0  0.0
1  0.0  0.0  0.0  0.0  0.0  1.0
2  0.0  0.0  0.0  0.0  0.0  1.0
3  0.0  0.0  0.0  0.0  0.0  1.0
4  0.0  0.0  0.0  0.0  0.0  1.0

[5 rows x 113 columns]
```

classify the normalized data

In [151]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

neigh = KNeighborsClassifier(n_neighbors=5)
y_pred1 = cross_val_predict(neigh, df_new, data_Y, cv=10)

# Classification Report
report_1 = classification_report(data_Y, y_pred1)
print(report_1)

# Accuracy
acc = accuracy_score(data_Y, y_pred1)
print('Accuracy:')
print(acc*100)
```

```
              precision    recall  f1-score   support

           0       0.97      0.68      0.80      1000
           1       0.76      0.98      0.85      1000

    accuracy                           0.83      2000
   macro avg       0.86      0.83      0.83      2000
weighted avg       0.86      0.83      0.83      2000

Accuracy:
83.25
```

find the best value of k for this data

In [152]:
```
from sklearn.model_selection import GridSearchCV

k_range = list(range(1, 31))
param_grid = dict(n_neighbors = k_range)

grid = GridSearchCV(neigh, param_grid, cv=10, scoring='accuracy')
grid.fit(df_new, data_Y)

# Print out the best value for each of the tested parameters.
print(grid.best_params_)
```

```
{'n_neighbors': 2}
```

Display the accuracy, precision, and recall of a KNN classifier

In [153]:
```
neigh2 = KNeighborsClassifier(n_neighbors=2)

# Run a 10-fold cross validation
better_scores = cross_val_score(neigh2, df_new, data_Y, cv=10)

# display accuracy
print('Accuracy:')
print(better_scores.mean()*100, '\n')

y_pred2 = cross_val_predict(neigh2, df_new, data_Y, cv=10)

# Classification Report
report_2 = classification_report(data_Y, y_pred2)
print(report_2)
```

```
Accuracy:
84.85

              precision    recall  f1-score   support

           0       0.90      0.78      0.84      1000
           1       0.81      0.92      0.86      1000

    accuracy                           0.85      2000
   macro avg       0.85      0.85      0.85      2000
weighted avg       0.85      0.85      0.85      2000
```

report the final accuarcy of your KNN model

In [154]:
```
# nested cross-validation
scores_2 = cross_val_score(grid, df_new, data_Y, cv=5)

# accuracy of scores
print("Accuracy:", scores_2.mean()*100)
```

```
Accuracy: 84.65
```

dimensionality reduction with PCA before running KNN.

In [155]:
```
# Define a pipeline to search for the best combination of
# PCA dimensions and n_neighbors.
import sklearn as sk
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline

#crate a scaler
scaler = sk.preprocessing.MinMaxScaler()

#create a PCA
pca = PCA()

#create a KNN classifier
knn = KNeighborsClassifier()

#create a pipeline that does scaling, then PCA, then KNN
pipe = Pipeline(steps=[('scaler', scaler), ('pca', pca), ('knn', knn)])

#Set up the parameters you want to tune for each of your pipeline steps
#Parameters of pipelines can be set using '__' separated parameter names:
param_grid = {
    'pca__n_components': list(range(1, 19)), #find how many principal componen
et to keep
    'knn__n_neighbors': list(range(1, 30)),  #find the best value of k
}


# pass the pipeline and the parameters into a GridSearchCV with a 5-fold cross
validation
my_grid = GridSearchCV(pipe, param_grid, cv=5)

# call fit() on the GridSearchCV and pass in the unscaled data (X_values, Y_va
lues)
my_grid.fit(data_X, data_Y)

# print out the best_score_ and best_params_ from the GridSearchCV
print("Best score: ", my_grid.best_score_, '\n')
print("Best parameters set: ", my_grid.best_params_)
```

```
Best score:  0.901

Best parameters set:  {'knn__n_neighbors': 21, 'pca__n_components': 8}
```

report accuracy

In [170]:
```
score_3 = cross_val_score(my_grid, data_X, data_Y, cv=5)

#accuracy of scores
print("Accuracy:", score_3.mean()*100)
```

```
Accuracy: 89.85
```

SVM Classifier:

In order to use an SVM classifier with categorical variables as features, we must encode them in some manner. The chosen method of encoding these variables is one-hot encoding, giving each instance of a variable a column in the dataset and a value of 1 if it is true. The Pandas function get_dummies allows us to add these dummy variables to the data.

In [156]:
```
# your code goes here
data_Y = one_encoded_data['class_Adoption']
data_X = one_encoded_data.drop(['class_Adoption'], axis = 1)
print(data_X.shape)
print(data_Y.shape)
print(data_X.head)
```

```
(2000, 113)
(2000,)
<bound method NDFrame.head of        animal_type_Bird  animal_type_Cat  anima
l_type_Dog  animal_type_Other  \
0                     0               1                0                 0
1                     0               0                1                 0
2                     0               0                1                 0
4                     0               1                0                 0
5                     0               1                0                 0
...                 ...             ...              ...               ...
12159                 0               0                0                 1
12183                 0               0                0                 1
12210                 0               0                0                 1
12212                 0               1                0                 0
12228                 0               0                1                 0

       sex_Intact Female  sex_Intact Male  sex_Neutered Male  \
0                      0                0                  1
1                      0                0                  1
2                      0                0                  1
4                      0                0                  1
5                      0                0                  1
...                  ...              ...                ...
12159                  0                0                  0
12183                  0                0                  0
12210                  0                0                  0
12212                  0                1                  0
12228                  0                1                  0

       sex_Spayed Female  sex_Unknown  breed_American Bulldog Mix  ...  \
0                      0            0                           0  ...
1                      0            0                           0  ...
2                      0            0                           0  ...
4                      0            0                           0  ...
5                      0            0                           0  ...
...                  ...          ...                         ...  ...
12159                  0            1                           0  ...
12183                  0            1                           0  ...
12210                  0            1                           0  ...
12212                  0            0                           0  ...
12228                  0            0                           0  ...

       color_White/Black  color_White/Blue  color_White/Brown  \
0                      0                 0                  0
1                      0                 0                  1
2                      0                 0                  0
4                      0                 0                  0
5                      0                 0                  0
...                  ...               ...                ...
12159                  0                 0                  0
12183                  0                 0                  0
12210                  0                 0                  0
12212                  0                 0                  0
12228                  0                 0                  0

       color_White/Brown Brindle  color_White/Brown Tabby  color_White/Gray
\
```

```
0                          0                      0                 0
1                          0                      0                 0
2                          0                      0                 0
4                          0                      0                 0
5                          0                      0                 0
...                      ...                    ...               ...
12159                      0                      0                 0
12183                      0                      0                 0
12210                      0                      0                 0
12212                      0                      0                 0
12228                      0                      0                 0

       color_White/Tan  color_Yellow  age_range_Adult  age_range_Young
0                    0             0                1                0
1                    0             0                0                1
2                    0             0                0                1
4                    0             0                0                1
5                    0             0                0                1
...                ...           ...              ...              ...
12159                0             0                0                1
12183                0             0                0                1
12210                0             0                0                1
12212                0             0                1                0
12228                0             0                1                0

[2000 rows x 113 columns]>
```

SVM and pipeline

In [160]:
```
# your code goes here
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score

scaler = StandardScaler()

clf = svm.SVC(gamma='auto')

pipe = Pipeline([('scaler', scaler), ('svm', clf)])

scores = cross_val_score(pipe, data_X, data_Y, cv=5)
print("Accuracy:", scores.mean()*100)
```

```
Accuracy: 87.65
```

Find best Kernel

In [161]:
```
# for the 'svm' part of the pipeline, tune the 'kernel' hyperparameter
param_grid = {'svm__kernel': ['linear', 'rbf', 'poly', 'sigmoid']}

# your code goes here
from sklearn.model_selection import GridSearchCV

# Takes in the pipeline as the estimator and runs a 5-fold-CV
grid_search = GridSearchCV(pipe, param_grid, cv=5, scoring='accuracy')

# Call fit to find the best kernel
grid_search.fit(data_X, data_Y)

# Printing out the best kernel
print(grid_search.best_params_)
```

```
{'svm__kernel': 'sigmoid'}
```

Accuracy of best Kernel

In [162]:
```
# your code goes here
scores = cross_val_score(grid_search, data_X, data_Y, cv=5)
print("Accuracy:", scores.mean()*100)
```

```
Accuracy: 90.05000000000001
```

Deploying the model

In [164]:
```
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score

# your code goes here
scaler = StandardScaler()

clf = MLPClassifier()

pipe = Pipeline([('scaler', scaler), ('mlp', clf)])

param_grid = {
    'mlp__hidden_layer_sizes': list(range(10, 70, 10)), #find the best hidden
 layer size
    'mlp__activation': ['logistic', 'tanh', 'relu'],  #find the best activatio
n functions
}

# pass the pipeline and the parameters into a GridSearchCV with a 5-fold cross
validation
grid_search = GridSearchCV(pipe, param_grid, cv=5, scoring='accuracy')

grid_search.fit(data_X, data_Y)

# Printing out the best parameters
print(grid_search.best_params_)

#set this final_model to your final model
final_model = grid_search

filename = 'finalized_model.sav'
pickle.dump(final_model, open(filename, 'wb'))
```

```
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
```

```
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
```

```
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
```

```
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
```

```
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
```

```
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
{'mlp__activation': 'logistic', 'mlp__hidden_layer_sizes': 20}
```

```
/Users/jesse/opt/anaconda3/lib/python3.7/site-packages/sklearn/neural_networ
k/multilayer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

We took one animal we knew was adopted, and one animal we knew that died to test the model

In [165]:
```
adopted_record = one_encoded_data.head(1)
dead_record = one_encoded_data.tail(1)
print("Adopted Record", adopted_record['class_Adoption'])
print("Dead_Record", dead_record['class_Adoption'])
adop_predicted_encoded_data = adopted_record.drop(columns = ['class_Adoption'
])
dead_predicted_encoded_data = dead_record.drop(columns = ['class_Adoption'])
```

```
Adopted Record 0    1
Name: class_Adoption, dtype: uint8
Dead_Record 12228    0
Name: class_Adoption, dtype: uint8
```

Animal we know that was adopted

In [166]:
```
adop_predicted_encoded_data.head()
```

Out[166]:

	animal_type_Bird	animal_type_Cat	animal_type_Dog	animal_type_Other	sex_Intact Female	sex_Intac Mal
0	0	1	0	0	0	(

1 rows × 113 columns

Animal we know that went to animal heaven

In [167]:
```
dead_predicted_encoded_data.head()
```

Out[167]:

	animal_type_Bird	animal_type_Cat	animal_type_Dog	animal_type_Other	sex_Intact Female	sex_
12228	0	0	1	0	0	

1 rows × 113 columns

Putting our dataframes into lists to test the classifier

In [168]:
```
adopt_list =[]
dead_list = []

# Iterate over each row
for i in range((adop_predicted_encoded_data.shape[0])):

    # Using iloc to access the values of
    # the current row denoted by "i"
    adopt_list.append(list(adop_predicted_encoded_data.iloc[i, :]))
    dead_list.append(list(dead_predicted_encoded_data.iloc[i, :]))

# Print the list
print(adopt_list)
print(dead_list)
```

```
[[0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0]]
[[0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0]]
```

Using our trained, saved classifier to classify an animal as adopted or "no longer with us"

In [169]:
```
# some time later...

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))

# your code goes here
labels = ["Went to a good home", "Sadly went on a vacation"]

record = {"Adopt":adopt_list, "Dead":dead_list}
for value in record.values():
    pred = loaded_model.predict(value)

    # Decide whether its adopted or going to a farm
    print("Prediction:  ")
    print(labels[0]) if pred == 1 else print(labels[1])
```

```
Prediction:
Went to a good home
Prediction:
Sadly went on a vacation
```

As you can see, the animal that actually was adopted was correctly classified as being adopted, and the animal that perished was correctly classified as "leaving this realm"

Outcome/Results:

These models could be used to predict if an incoming animal to a shelter has an increased risk of being euthanized v. being adopted. Using this information, shelter leaders could focus advertising efforts for certain breeds of pets or contact breed-specific rescue groups to help them increase adoption levels for certain animal types or breeds.

In [ ]: