In [13]:
```
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df_horse = pd.read_csv("horse.csv")
df_horse.head()
```

Out[4]:

	surgery	age	hospital_number	rectal_temp	pulse	respiratory_rate	temp_of_extremities	pe
0	no	adult	530101	38.5	66.0	28.0	cool	
1	yes	adult	534817	39.2	88.0	20.0	NaN	
2	no	adult	530334	38.3	40.0	24.0	normal	
3	yes	young	5290409	39.1	164.0	84.0	cold	
4	no	adult	530255	37.3	104.0	35.0	NaN	

5 rows × 28 columns

In [7]:
```
# 1. Let's attempt to predict the survival of a horse based on various observe
d medical conditions.
df_horse["outcome"]
```

Out[7]:
```
0            died
1      euthanized
2           lived
3            died
4            died
          ...
294    euthanized
295    euthanized
296          died
297         lived
298    euthanized
Name: outcome, Length: 299, dtype: object
```

In [8]:
```
# Load the data from 'horses.csv' and observe whether it contains missing values.
df_horse.isnull()
```

Out[8]:

	surgery	age	hospital_number	rectal_temp	pulse	respiratory_rate	temp_of_extremities
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	True
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	True
...	...	...	...	...	...	...	...
294	False	False	False	True	False	False	False
295	False	False	False	False	False	False	False
296	False	False	False	False	False	False	False
297	False	False	False	False	False	False	False
298	False	False	False	False	False	False	True

299 rows × 28 columns

In [12]:
```
# 2. This dataset contains many categorical features, replace them with label encoding.
# [Hint: Refer to get_dummies methods in pandas dataframe or Label encoder in scikit-learn]
Y = df_horse['outcome']
X = df_horse.drop(['outcome'], axis=1)

X = pd.get_dummies(X)
X.head()
```

Out[12]:

	hospital_number	rectal_temp	pulse	respiratory_rate	nasogastric_reflux_ph	packed_cell_volu
0	530101	38.5	66.0	28.0	NaN	4
1	534817	39.2	88.0	20.0	NaN	5
2	530334	38.3	40.0	24.0	NaN	3
3	5290409	39.1	164.0	84.0	5.0	4
4	530255	37.3	104.0	35.0	NaN	7

5 rows × 67 columns

In [26]:
```
# 3. Replace the missing values by the most frequent value in each column.
#imp = SimpleImputer(missing_values = np.nan, strategy='mean')
#imp = imp.fit(df_horse)  #fitting data into imputer object
#df_horse = imp.transform(df_horse)   # imputing the data

X = X.apply(lambda x:x.fillna(x.value_counts().index[0]))
```

In [27]:
```
# 4.Fit a decision tree classifier and observe the accuracy.
dec_tree = DecisionTreeClassifier()
dec_tree.fit(X, Y)
predicted_outcome = dec_tree.predict(X)
metrics.accuracy_score(predicted_outcome, Y)
```

Out[27]: 1.0

In [29]:
```
# 5.Fit a random forest classifier and observe the accuracy
random_forest = RandomForestClassifier()
random_forest.fit(X,Y)
predicted_outcome = random_forest.predict(X)
metrics.accuracy_score(predicted_outcome, Y)
```

Out[29]: 1.0

In [ ]: