0.) Import and Clean data

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import plot_tree
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysi
```

In [3]:
```
df = pd.read_csv("bank.csv", sep = ";")
```

In [4]:
```
df.head()
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	

In [5]:
```
df.columns
```

Out[5]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housi
ng',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pday
s',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [6]:

```
df = df.drop(["default", "pdays", "previous", "poutcome"], axis = 1)
df = pd.get_dummies(df, columns = ["loan", "job","marital","housing",'

```

In [7]:

```
df.head()
```

Out[7]:

	age	balance	day	duration	y	loan_yes	job_blue-collar	job_entrepreneur	job_housemaid
0	30	1787	19	79	no	0	0	0	0
1	33	4789	11	220	no	1	0	0	0
2	35	1350	16	185	no	0	0	0	0
3	30	1476	3	199	no	1	0	0	0
4	59	0	5	226	no	0	1	0	0

5 rows × 67 columns

In [8]:

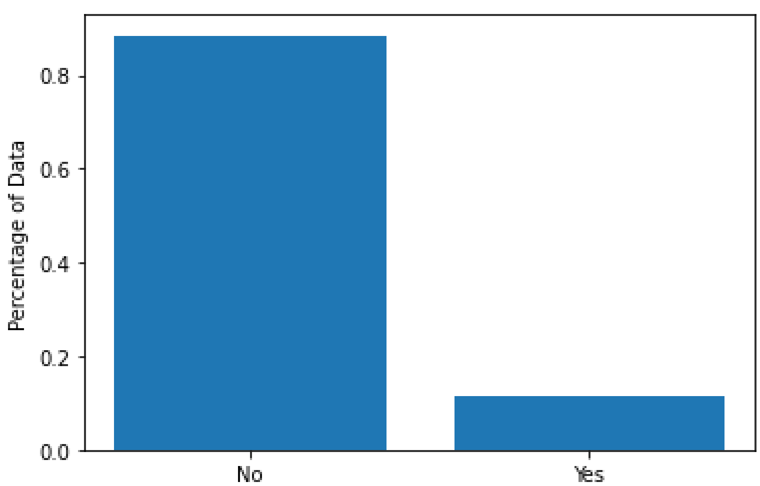
```
y = pd.get_dummies(df["y"], drop_first = True)
X = df.drop(["y"], axis = 1)
```

In [ ]:

```

```

In [9]:

```
obs = len(y)
plt.bar(["No","Yes"],[len(y[y.yes==0])/obs,len(y[y.yes==1])/obs])
plt.ylabel("Percentage of Data")
plt.show()
```



In [10]:
```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

scaler = StandardScaler().fit(X_train)

X_scaled = scaler.transform(X_train)
X_test = scaler.transform(X_test)

```

In [72]:
```
y_train.mean()
```

Out[72]:
```
yes    0.116625
dtype: float64
```

#1.) Based on the visualization above, use your expert opinion to transform the data based on what we learned this quarter

In [11]:
```
# ###############
# ###TRANSFORM###
# # ###############

# X_scaled = X_SMOTE
# y_train = y_SMOTE
```

2.) Build and visualize a decision tree of Max Depth 3. Show the confusion matrix.

In [12]:
```
dtree = DecisionTreeClassifier(max_depth = 3)
dtree.fit(X_scaled, y_train)
```

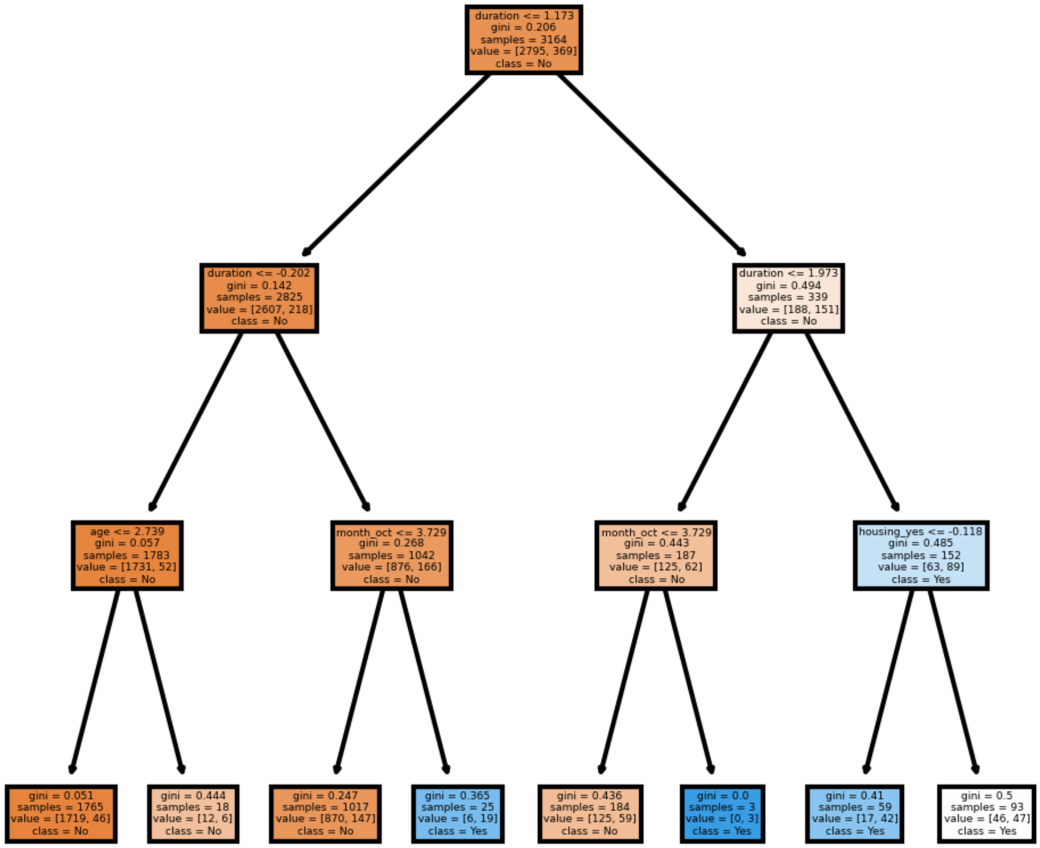
Out[12]:
```
DecisionTreeClassifier(max_depth=3)
```

In [13]:

```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(dtree, filled = True, feature_names = X.columns, class_names


#fig.savefig('imagename.png')
```

Out[13]:

```
[Text(0.5, 0.875, 'duration <= 1.173\ngini = 0.206\nsamples = 3164\nvalue
= [2795, 369]\nclass = No'),
 Text(0.25, 0.625, 'duration <= -0.202\ngini = 0.142\nsamples = 2825\nval
ue = [2607, 218]\nclass = No'),
 Text(0.125, 0.375, 'age <= 2.739\ngini = 0.057\nsamples = 1783\nvalue =
[1731, 52]\nclass = No'),
 Text(0.0625, 0.125, 'gini = 0.051\nsamples = 1765\nvalue = [1719, 46]\nc
lass = No'),
 Text(0.1875, 0.125, 'gini = 0.444\nsamples = 18\nvalue = [12, 6]\nclass
= No'),
 Text(0.375, 0.375, 'month_oct <= 3.729\ngini = 0.268\nsamples = 1042\nva
lue = [876, 166]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.247\nsamples = 1017\nvalue = [870, 147]\nc
lass = No'),
 Text(0.4375, 0.125, 'gini = 0.365\nsamples = 25\nvalue = [6, 19]\nclass
= Yes'),
 Text(0.75, 0.625, 'duration <= 1.973\ngini = 0.494\nsamples = 339\nvalue
= [188, 151]\nclass = No'),
 Text(0.625, 0.375, 'month_oct <= 3.729\ngini = 0.443\nsamples = 187\nval
ue = [125, 62]\nclass = No'),
 Text(0.5625, 0.125, 'gini = 0.436\nsamples = 184\nvalue = [125, 59]\ncla
ss = No'),
 Text(0.6875, 0.125, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]\nclass = Ye
s'),
 Text(0.875, 0.375, 'housing_yes <= -0.118\ngini = 0.485\nsamples = 152\n
value = [63, 89]\nclass = Yes'),
 Text(0.8125, 0.125, 'gini = 0.41\nsamples = 59\nvalue = [17, 42]\nclass
= Yes'),
 Text(0.9375, 0.125, 'gini = 0.5\nsamples = 93\nvalue = [46, 47]\nclass =
Yes')]
```



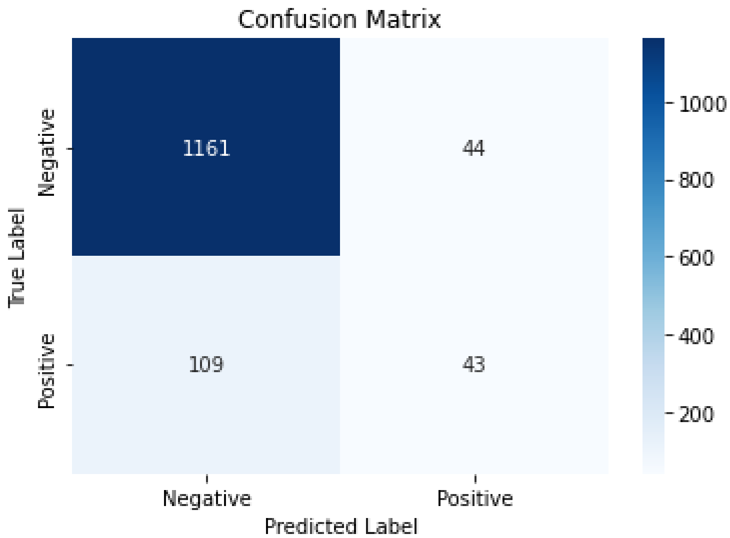
1b.) Confusion matrix on out of sample data. Visualize and store as variable

In [14]:

```
y_pred = dtree.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

In [15]:

```
class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_raw, annot=True, fmt='d', cmap='Blues', xticklabels=cla
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



3.) Use bagging on your descision tree

In [16]:

```
# X = X_train
# y = y_train

# Create and fit bagging classifier
dtree = DecisionTreeClassifier(max_depth=3)
bagging = BaggingClassifier(base_estimator=dtree, n_estimators=100, ma
bagging.fit(X_train, y_train)

# Predict and evaluate
y_pred = bagging.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

```

```
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\ensemble\_bagging.py:7
19: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)

Accuracy: 0.887988209285188

C:\Users\parzu\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but BaggingClassifier was fitted
with feature names
  warnings.warn(
```

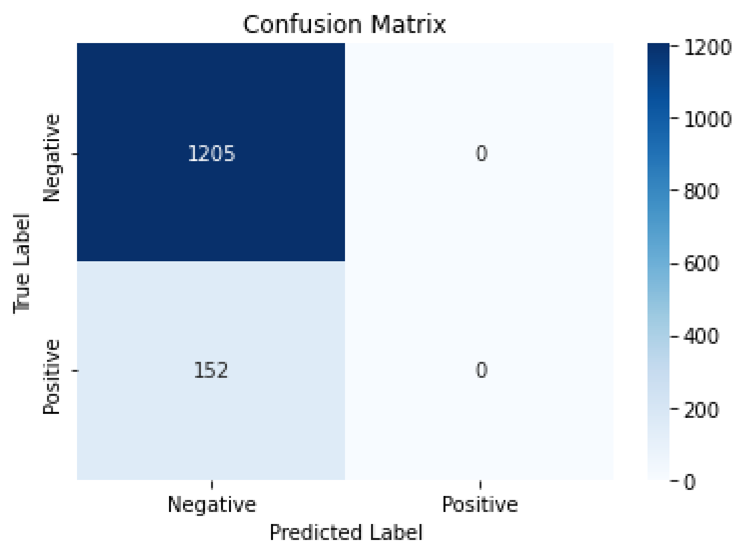
In [17]:

```
y_pred = bagging.predict(X_test)
y_true = y_test
cm_bag = confusion_matrix(y_true, y_pred)
```

```
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but BaggingClassifier was fitted
with feature names
  warnings.warn(
```

In [18]:

```
class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_bag, annot=True, fmt='d', cmap='Blues', xticklabels=cla
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



4.) Boost your tree

In [19]:

```
from sklearn.ensemble import AdaBoostClassifier
```

In [20]:

```
dtree = DecisionTreeClassifier(max_depth=3)
adaboost = AdaBoostClassifier(base_estimator=dtree, n_estimators=50, l

adaboost.fit(X_train, y_train)

y_pred = adaboost.predict(X_test)
```

```
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\utils\validation.py:99
3: DataConversionWarning: A column-vector y was passed when a 1d array wa
s expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but AdaBoostClassifier was fitte
d with feature names
  warnings.warn(
```

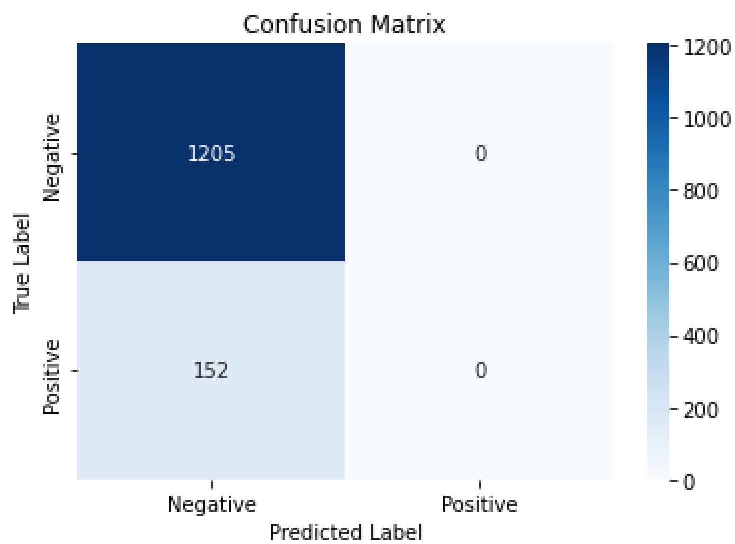
In [21]:

```
y_pred = adaboost.predict(X_test)
y_true = y_test
cm_boost = confusion_matrix(y_true, y_pred)
```

```
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but AdaBoostClassifier was fitte
d with feature names
  warnings.warn(
```

In [22]:

```
class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_boost, annot=True, fmt='d', cmap='Blues', xticklabels=c
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [23]:

```
boosting_accuracy = round(accuracy_score(y_test, y_pred),3)
boosting_accuracy
```

Out[23]: 0.888

5.) Create a superlearner with at least 5 base learner models. Use a logistic reg for your metalearner. Interpret your coefficients and save your CM.

In [25]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
####IMPORT MORE BASE LEARNERS####

from mlens.ensemble import SuperLearner
```

In [109]:

```
# # ### SET YOUR BASE LEARNERS
# base_estimaters = [
#    LogisticRegression()
# # ]

# # super_learner = SuperLearner ()
# # super_learner.add(base_estimaters )

# # ### FIT TO TRAINING DATA
# # super_learner.fit(X_train, y_train)
# # ### GET base_predictions

# # base_predictions = super_learner.fit(X_train)
```

In [ ]:

```

```

In [26]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoosting(
from mlens.ensemble import SuperLearner
from sklearn.metrics import confusion_matrix
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysi
from sklearn.svm import SVC
```

In [29]:

```
base_estimators = [ RandomForestClassifier(random_state=101), Gradient

super_learner = SuperLearner(folds=10, random_state=42)
super_learner.add(base_estimators)
super_learner.fit(X_scaled, y_train)

base_predictions = super_learner.predict(X_test)

```

```
  return self._fit(X, y)
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\neighbors\_classifi
cation.py:198: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  return self._fit(X, y)
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\neighbors\_classifi
cation.py:198: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  return self._fit(X, y)
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\neighbors\_classifi
cation.py:198: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  return self._fit(X, y)
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\utils\validation.p
y:993: DataConversionWarning: A column-vector y was passed when a 1d a
rray was expected. Please change the shape of y to (n_samples, ), for
example using ravel().
```

In [119]:

```
### TRAIN YOUR METALEARNER
```

In [30]:

```
# Preprocess X_test
base_predictions_test = super_learner.predict(X_test)
```

In [31]:

```
y_train.values.reshape(len(y_train),).shape
```

Out[31]:

```
(3164,)
```

In [32]:

```
base_predictions
```

Out[32]:

```
array([[0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       ...,
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.]], dtype=float32)
```

In [34]:
```
X_scaled.shape
```
Out[34]: (3164, 66)

In [35]:
```
X_test.shape
```
Out[35]: (1357, 66)

In [36]:
```
Log_reg = LogisticRegression(fit_intercept = False).fit(X_scaled,y_tra
y_pred = Log_reg.predict(X_test)
```

In [ ]:
```
### INTERPRET COEFFICIENTS
```

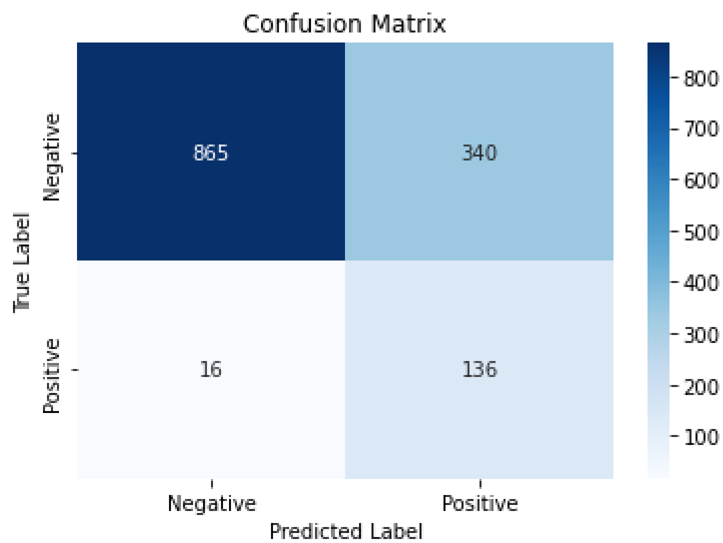
In [41]:
```
Log_reg.coef_
```
Out[41]:
```
array([[ 2.99482647e-02, -2.07906128e-02,  5.25933838e-02,
         7.23533812e-01, -5.74514805e-02, -8.04015134e-02,
        -5.42396930e-02, -1.36875158e-02, -3.83995673e-02,
         3.60674360e-02, -1.61779931e-02, -3.71929050e-02,
         2.50415602e-02, -4.36219942e-02, -6.79631027e-02,
         1.06432242e-02, -7.97132721e-02, -4.15474785e-02,
        -6.07221985e-02,  1.89421162e-02, -1.86813553e-01,
        -6.73457975e-02, -3.66996359e-02, -1.10929490e-03,
        -3.06184205e-02, -1.81181958e-02, -1.57524856e-02,
        -1.77253008e-02,  1.63732997e-03, -2.19696958e-02,
        -3.37525124e-03, -8.23646052e-03, -1.00219752e-03,
        -9.34799995e-03, -7.44656399e-03, -9.33704216e-03,
         9.92128396e-03,  9.27480760e-03, -1.23920699e-02,
        -4.28404898e-02,  3.83989939e-03, -1.80772529e-02,
        -9.99484937e-03, -3.81974867e-04,  5.32158876e-03,
        -1.97283851e-02,  7.82176181e-04,  3.66060762e-03,
         9.54145314e-03,  1.71400261e-03,  5.56776161e-03,
         5.97370942e-03, -9.30571705e-02,  6.06659618e-02,
         1.21674990e-03, -5.88368805e-02, -1.33298178e-01,
         4.32922016e-02,  1.48996578e-01, -9.96601459e-02,
        -9.39024136e-02,  2.39218625e-01,  7.58023383e-02,
         2.01032260e-02,  7.36824974e-02, -2.70084716e-02]])
```

Each component of this array reflects the weight or coefficient of the related feature in the logistic regression model.

In [43]:

```
cm2 = confusion_matrix(y_true, y_pred)

accuracy = accuracy_score(y_true, y_pred)
print("Metal Learner Accuracy:", accuracy)

class_labels = ['Negative', 'Positive']

sns.heatmap(cm2, annot=True, fmt='d', cmap='Blues', xticklabels=class_
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
Metal Learner Accuracy: 0.737656595431098
```



6.) Create a bar chart comparing decision tree, bagged, boosted and super learner Sensitivities (Out of Sample)

In [48]:

```
sensitivity = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )

specificity = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
```

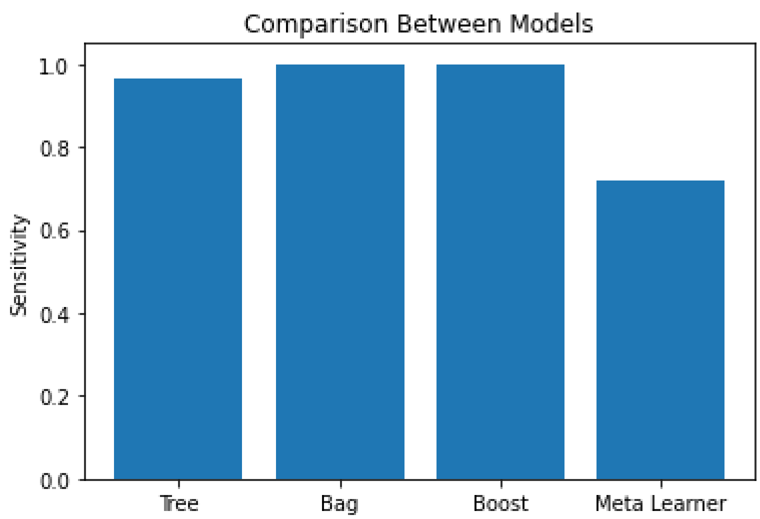
```
Sensitivity :  0.7178423236514523
Specificity :  0.8947368421052632
```

In [52]:

```
sen_raw = cm_raw[0,0]/(cm_raw[0,0]+cm_raw[0,1])
spec_raw = cm_raw[1,1]/(cm_raw[1,0]+cm_raw[1,1])

# Bagging
sen_bag = cm_bag[0,0]/(cm_bag[0,0]+cm_bag[0,1])
spec_bag = cm_bag[1,1]/(cm_bag[1,0]+cm_bag[1,1])

# Boosting
sen_boost = cm_boost[0,0]/(cm_boost[0,0]+cm_boost[0,1])
spec_boost = cm_boost[1,1]/(cm_boost[1,0]+cm_boost[1,1])

# meta Learner
sen_SL = cm2[0,0]/(cm2[0,0]+cm2[0,1])
spec_SL = cm2[1,1]/(cm2[1,0]+cm2[1,1])
```

In [57]:

```
plt.title("Comparison Between Models")
plt.bar(["Tree","Bag","Boost","Meta Learner"],[sen_raw,sen_bag,sen_boo
plt.ylabel("Sensitivity")
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: