0.) Import and Clean data

In [39]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [40]:
```python
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
```

In [41]:
```python
df = pd.read_csv("bank.csv", delimiter=';')
```

In [42]:
```python
df.head()
```

Out[42]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	n
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	

5 rows × 21 columns

In [43]:
```python
df = df.drop(["default", "pdays",   "previous", "poutcome", "emp.var.rate", "cons.price.idx",   "cons.conf.idx",    "euribor3m",    "nr.employed"],
df = pd.get_dummies(df, columns = ["loan", "job","marital","housing","contact","day_of_week", "campaign", "month", "education"],drop_first = True)
```

In [44]:
```python
df.head()
```

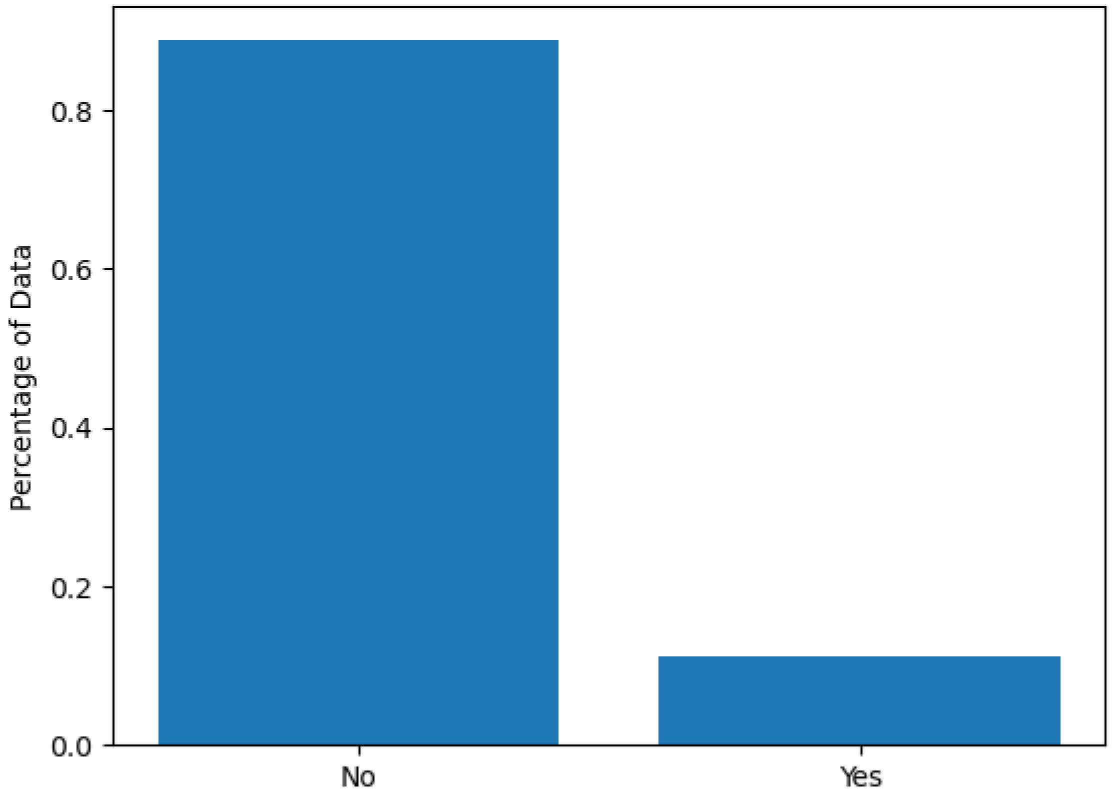
Out[44]:

	age	duration	y	loan_unknown	loan_yes	job_blue-collar	job_entrepreneur	job_housemaid	job_management	job_retired	...	month_nov	month_oct	month_sep	education_basic.6y	education_basic.!
0	56	261	no	False	False	False	False	True	False	False	...	False	False	False	False	Fal:
1	57	149	no	False	False	False	False	False	False	False	...	False	False	False	False	Fal:
2	37	226	no	False	False	False	False	False	False	False	...	False	False	False	False	Fal:
3	40	151	no	False	False	False	False	False	False	False	...	False	False	False	True	Fal:
4	56	307	no	False	True	False	False	False	False	False	...	False	False	False	False	Fal:

5 rows × 83 columns

In [45]:
```python
y = pd.get_dummies(df["y"], drop_first = True)
X = df.drop(["y"], axis = 1)
```

In [46]:
```
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 [47]:
```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler().fit(X_train)

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

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

In [48]:
```
###############
###TRANSFORM###
###############

#X_scaled = #???
# y_train = #???
```

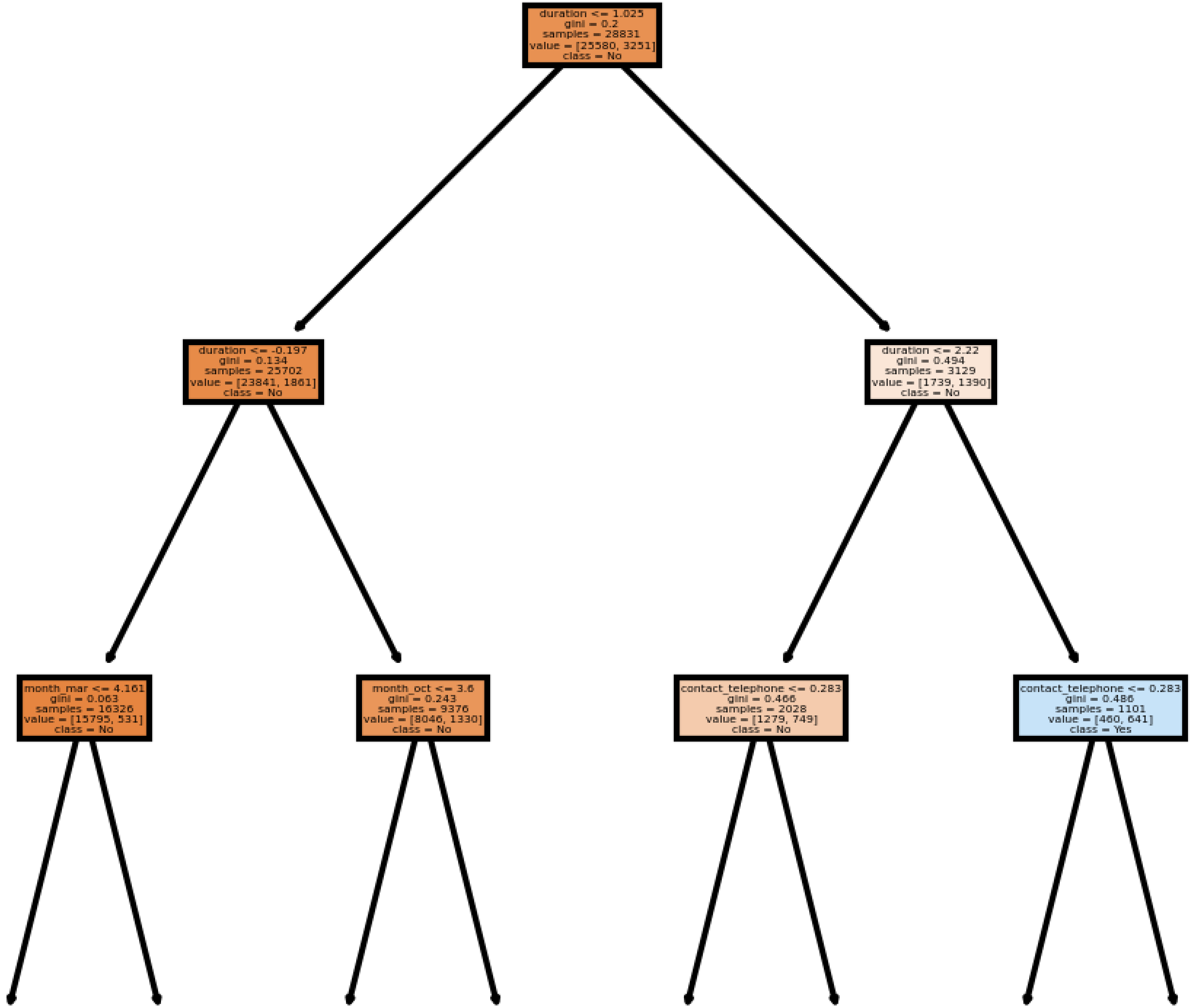
2.) Build and visualize a decision tree of Max Depth 3. Show the confusion matrix.

In [49]:
```
dtree_main = DecisionTreeClassifier(max_depth = 3)
dtree_main.fit(X_scaled, y_train)
```

Out[49]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

In [50]:
```python
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(dtree_main, filled = True, feature_names = list(X.columns), class_names=["No","Yes"])


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

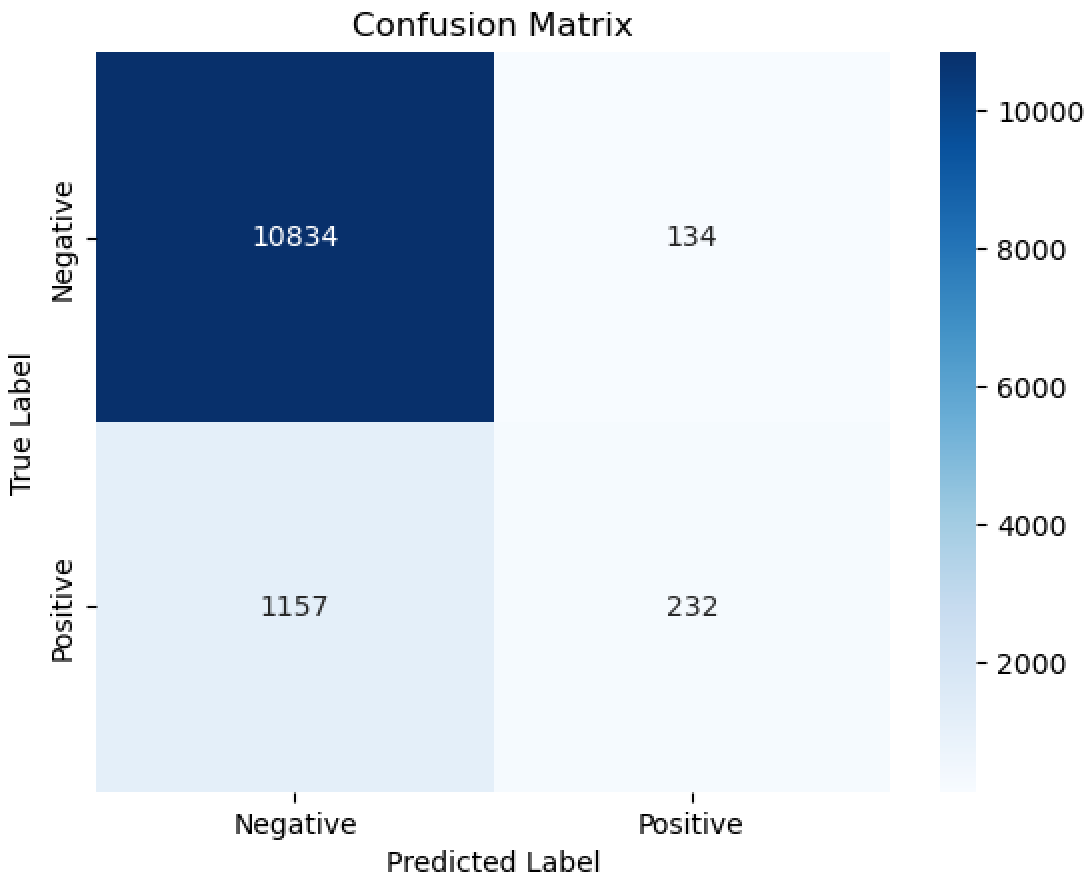
Out[50]:
```
[Text(0.5, 0.875, 'duration <= 1.025\ngini = 0.2\nsamples = 28831\nvalue = [25580, 3251]\nclass = No'),
 Text(0.25, 0.625, 'duration <= -0.197\ngini = 0.134\nsamples = 25702\nvalue = [23841, 1861]\nclass = No'),
 Text(0.125, 0.375, 'month_mar <= 4.161\ngini = 0.063\nsamples = 16326\nvalue = [15795, 531]\nclass = No'),
 Text(0.0625, 0.125, 'gini = 0.055\nsamples = 16102\nvalue = [15644, 458]\nclass = No'),
 Text(0.1875, 0.125, 'gini = 0.439\nsamples = 224\nvalue = [151, 73]\nclass = No'),
 Text(0.375, 0.375, 'month_oct <= 3.6\ngini = 0.243\nsamples = 9376\nvalue = [8046, 1330]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.227\nsamples = 9175\nvalue = [7980, 1195]\nclass = No'),
 Text(0.4375, 0.125, 'gini = 0.441\nsamples = 201\nvalue = [66, 135]\nclass = Yes'),
 Text(0.75, 0.625, 'duration <= 2.22\ngini = 0.494\nsamples = 3129\nvalue = [1739, 1390]\nclass = No'),
 Text(0.625, 0.375, 'contact_telephone <= 0.283\ngini = 0.466\nsamples = 2028\nvalue = [1279, 749]\nclass = No'),
 Text(0.5625, 0.125, 'gini = 0.49\nsamples = 1364\nvalue = [777, 587]\nclass = No'),
 Text(0.6875, 0.125, 'gini = 0.369\nsamples = 664\nvalue = [502, 162]\nclass = No'),
 Text(0.875, 0.375, 'contact_telephone <= 0.283\ngini = 0.486\nsamples = 1101\nvalue = [460, 641]\nclass = Yes'),
 Text(0.8125, 0.125, 'gini = 0.47\nsamples = 745\nvalue = [281, 464]\nclass = Yes'),
 Text(0.9375, 0.125, 'gini = 0.5\nsamples = 356\nvalue = [179, 177]\nclass = No')]
```


duration <= 1.025
gini = 0.2
samples = 28831
value = [25580, 3251]
class = No
duration <= -0.197
gini = 0.134
samples = 25702
value = [23841, 1861]
class = No
duration <= 2.22
gini = 0.494
samples = 3129
value = [1739, 1390]
class = No
month_mar <= 4.161
gini = 0.063
samples = 16326
value = [15795, 531]
class = No
month_oct <= 3.6
gini = 0.243
samples = 9376
value = [8046, 1330]
class = No
contact_telephone <= 0.283
gini = 0.466
samples = 2028
value = [1279, 749]
class = No
contact_telephone <= 0.283
gini = 0.486
samples = 1101
value = [460, 641]
class = Yes

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

In [51]:
```
y_pred = dtree_main.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

In [52]:
```
class_labels = ['Negative', 'Positive']

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



3.) Use bagging on your descision tree

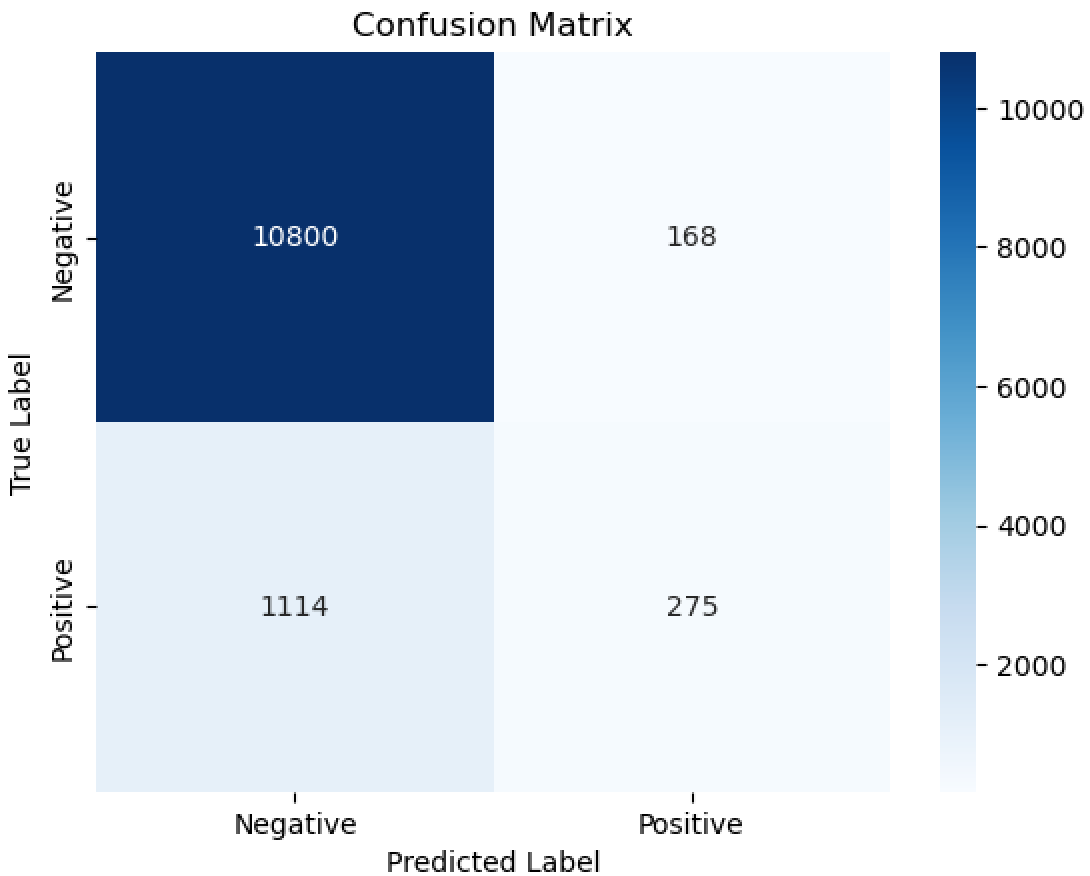
In [53]:
```
#Optimize on Max Depth...
dtree = DecisionTreeClassifier(max_depth = 3)
```

In [54]:
```
bagging = BaggingClassifier(estimator = dtree,
                 n_estimators = 100,
                 max_samples = .5,
                 max_features = 1.)
bagging.fit(X_scaled, y_train)

y_pred = bagging.predict(X_test)
```

```
/Users/frank/anaconda3/lib/python3.11/site-packages/sklearn/ensemble/_bagging.py:802: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [55]:
```
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

In [56]:
```
class_labels = ['Negative', 'Positive']

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



4.) Boost your tree

In [57]:
```
from sklearn.ensemble import AdaBoostClassifier
```

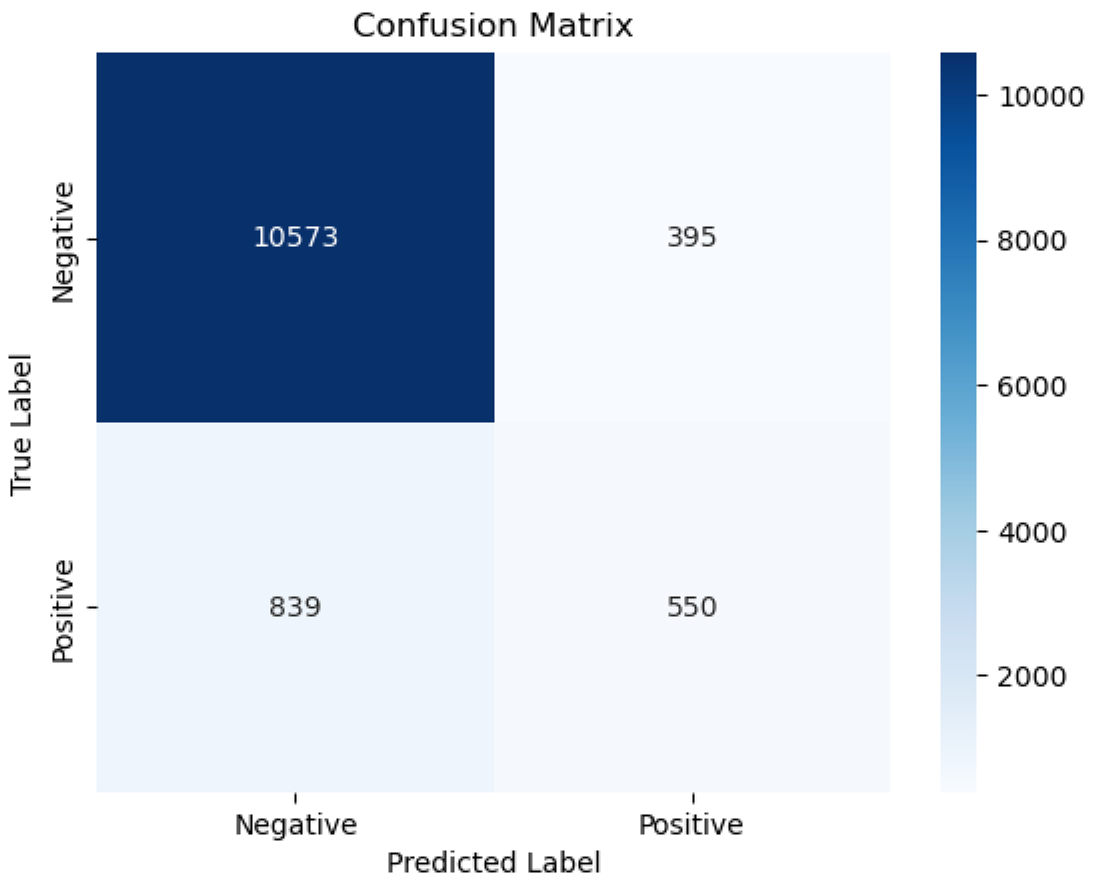
In [58]:
```
#Optimize on Max Depth...
dtree = DecisionTreeClassifier(max_depth = 3)
```

In [59]:
```
boost = AdaBoostClassifier(estimator = dtree)
boost.fit(X_scaled,y_train)

y_pred = boost.predict(X_test)
```

```
/Users/frank/anaconda3/lib/python3.11/site-packages/sklearn/utils/validation.py:1184: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [60]:
```
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

In [61]:
```
class_labels = ['Negative', 'Positive']

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



5.) Train a logistic regression on the decision Tree ,Boosted Tree ,Bagged Tree. Interpret coefficients and significance.

In [62]:
```
from sklearn.linear_model import LogisticRegression
```

In [63]:
```
X_base_learners = [bagging.predict(X_scaled),
                   boost.predict(X_scaled),
                   dtree_main.predict(X_scaled)]
X_base_learners = np.column_stack(X_base_learners)
```

In [64]:
```
super_learner = LogisticRegression()
```

In [65]:
```
super_learner.fit(X_base_learners,y_train)
super_learner.coef_
```

```
/Users/frank/anaconda3/lib/python3.11/site-packages/sklearn/utils/validation.py:1184: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[65]:
```
array([[0.65363619, 3.05832347, 0.08887617]])
```

In the logistic regression model trained on the predictions from three decision tree models (bagged, boosted, and a simple decision tree), the coefficients indicate the relative importance of each base learner's predictions. A higher coefficient value signifies a greater impact on the model's decisions. The boosting decision tree's predictions have the highest positive influence, followed by the bagged tree, while the simple decision tree has the least impact on the logistic regression outcome. These coefficients help understand how each base learner contributes to the ensemble's predictive power, with statistical significance tests needed to confirm the reliability of these interpretations.

In [ ]: