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
```

In [3]:
```
#drive.mount('/content/gdrive/', force_remount = True)
```

In [4]:
```
#df = pd.read_csv('/Users/akshitakhajuria/Downloads/bank-additional-full (1)
df = pd.read_csv('~/Downloads/bank-additional-full.csv', sep=';')
```

In [5]:
```
df
```

Out[5]:

	age	job	marital	education	default	housing	loan	contact	mont
0	56	housemaid	married	basic.4y	no	no	no	telephone	ma
1	57	services	married	high.school	unknown	no	no	telephone	ma
2	37	services	married	high.school	no	yes	no	telephone	ma
3	40	admin.	married	basic.6y	no	no	no	telephone	ma
4	56	services	married	high.school	no	no	yes	telephone	ma
...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	no
41184	46	blue-collar	married	professional.course	no	no	no	cellular	no
41185	56	retired	married	university.degree	no	yes	no	cellular	no
41186	44	technician	married	professional.course	no	no	no	cellular	no
41187	74	retired	married	professional.course	no	yes	no	cellular	no

41188 rows × 21 columns

In [6]:
```
df = df.drop(["default", "pdays",       "previous",     "poutcome",     "emp
df = pd.get_dummies(df, columns = ["loan", "job","marital","housing","contac
```

In [7]:
```
df.head()
```

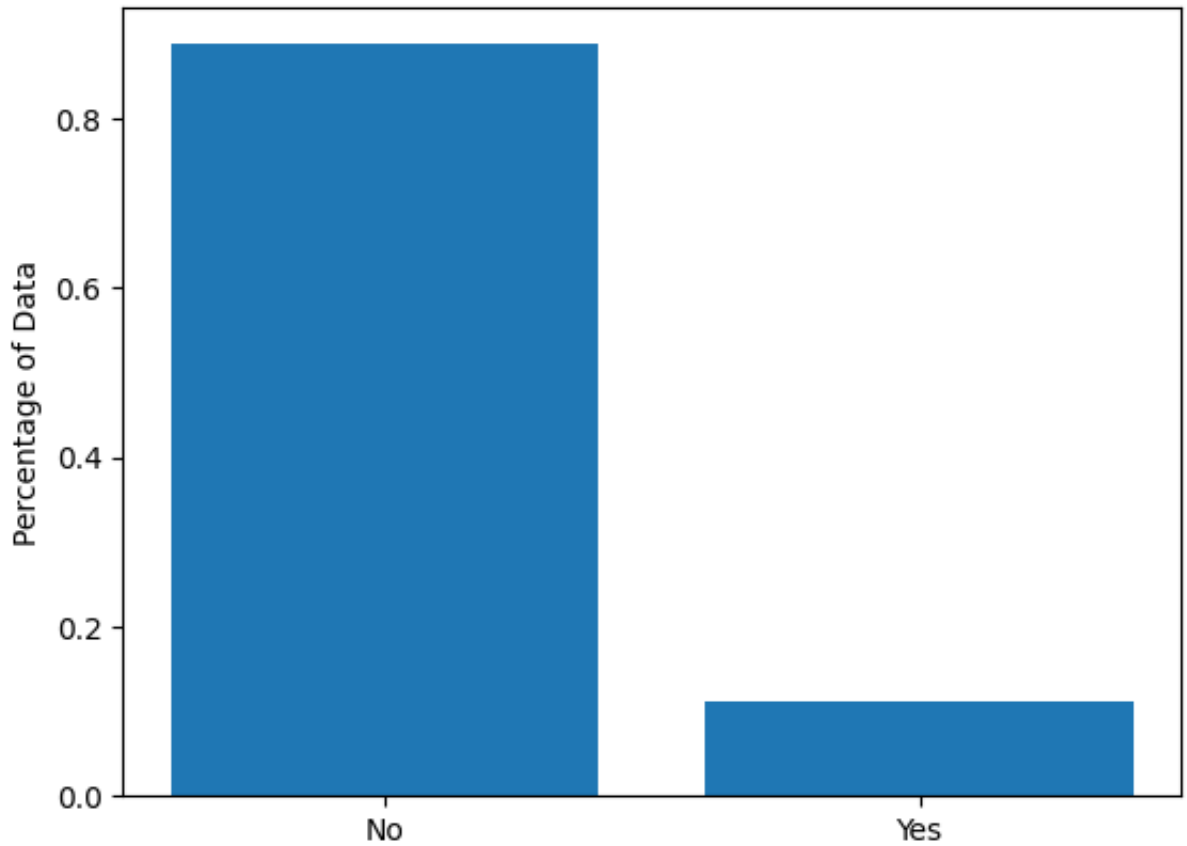
Out[7]:

	age	duration	y	loan_unknown	loan_yes	job_blue-collar	job_entrepreneur	job_housemaid
0	56	261	no	False	False	False	False	True
1	57	149	no	False	False	False	False	False
2	37	226	no	False	False	False	False	False
3	40	151	no	False	False	False	False	False
4	56	307	no	False	True	False	False	False

5 rows × 83 columns

In [8]:
```
y = pd.get_dummies(df["y"], drop_first = True)
X = df.drop(["y"], axis = 1)
```

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 [11]:
```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran

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 [12]:
```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X_scaled,y_train = oversample.fit_resample(X_scaled,y_train)
```

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

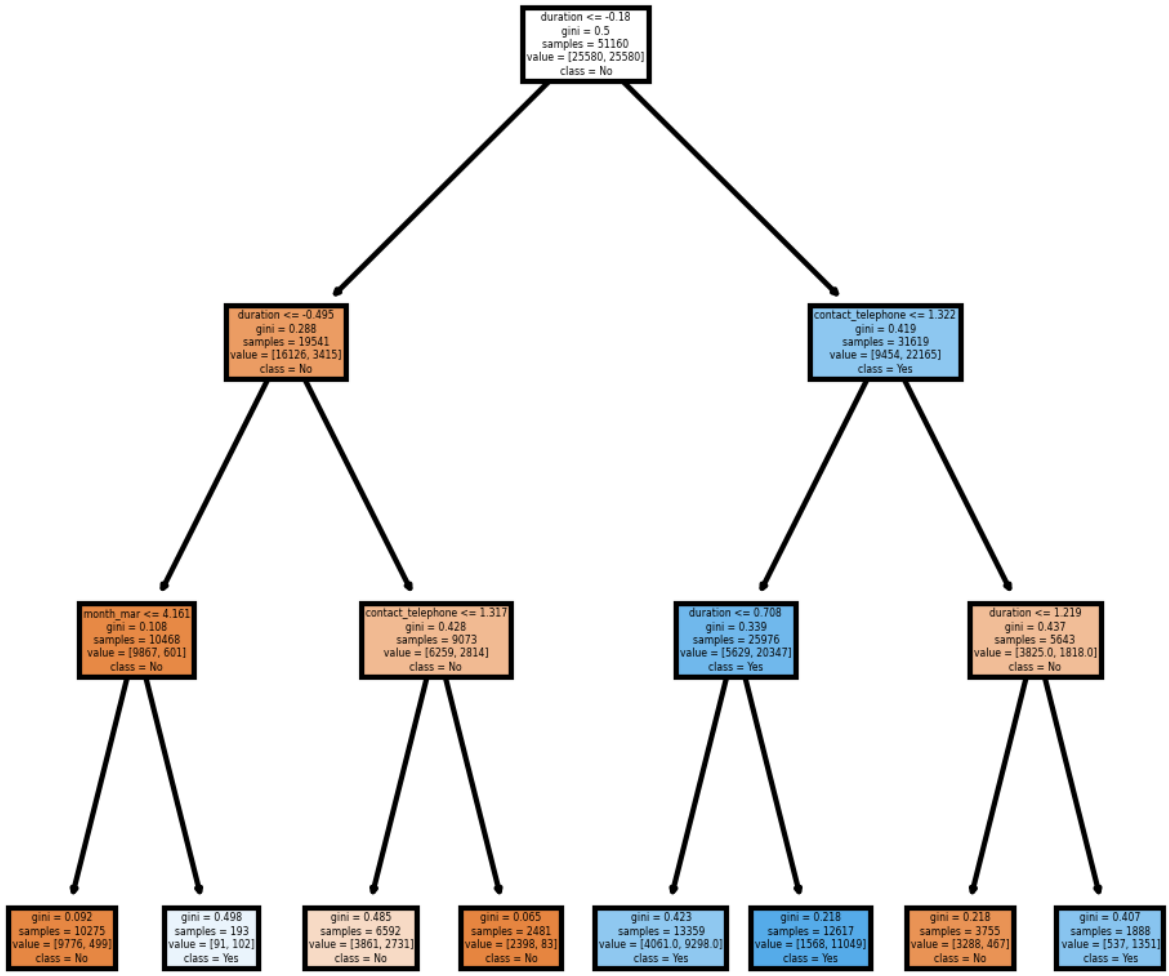
In [61]:
```
dtree_main = DecisionTreeClassifier(max_depth = 3)
dtree_main.fit(X_scaled, y_train)
```

Out[61]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

In [62]:
```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(dtree, filled = True, feature_names = X.columns, class_names=["No"


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

Out[62]:
```
[Text(0.5, 0.875, 'duration <= -0.18\ngini = 0.5\nsamples = 51160\nvalue = [
25580, 25580]\nclass = No'),
 Text(0.25, 0.625, 'duration <= -0.495\ngini = 0.288\nsamples = 19541\nvalue
= [16126, 3415]\nclass = No'),
 Text(0.125, 0.375, 'month_mar <= 4.161\ngini = 0.108\nsamples = 10468\nvalu
e = [9867, 601]\nclass = No'),
 Text(0.0625, 0.125, 'gini = 0.092\nsamples = 10275\nvalue = [9776, 499]\ncl
ass = No'),
 Text(0.1875, 0.125, 'gini = 0.498\nsamples = 193\nvalue = [91, 102]\nclass
= Yes'),
 Text(0.375, 0.375, 'contact_telephone <= 1.317\ngini = 0.428\nsamples = 907
3\nvalue = [6259, 2814]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.485\nsamples = 6592\nvalue = [3861, 2731]\ncl
ass = No'),
 Text(0.4375, 0.125, 'gini = 0.065\nsamples = 2481\nvalue = [2398, 83]\nclas
s = No'),
 Text(0.75, 0.625, 'contact_telephone <= 1.322\ngini = 0.419\nsamples = 3161
9\nvalue = [9454, 22165]\nclass = Yes'),
 Text(0.625, 0.375, 'duration <= 0.708\ngini = 0.339\nsamples = 25976\nvalue
= [5629, 20347]\nclass = Yes'),
 Text(0.5625, 0.125, 'gini = 0.423\nsamples = 13359\nvalue = [4061.0, 9298.0
]\nclass = Yes'),
 Text(0.6875, 0.125, 'gini = 0.218\nsamples = 12617\nvalue = [1568, 11049]\n
class = Yes'),
 Text(0.875, 0.375, 'duration <= 1.219\ngini = 0.437\nsamples = 5643\nvalue
= [3825.0, 1818.0]\nclass = No'),
 Text(0.8125, 0.125, 'gini = 0.218\nsamples = 3755\nvalue = [3288, 467]\ncla
ss = No'),
 Text(0.9375, 0.125, 'gini = 0.407\nsamples = 1888\nvalue = [537, 1351]\ncla
ss = Yes')]
```

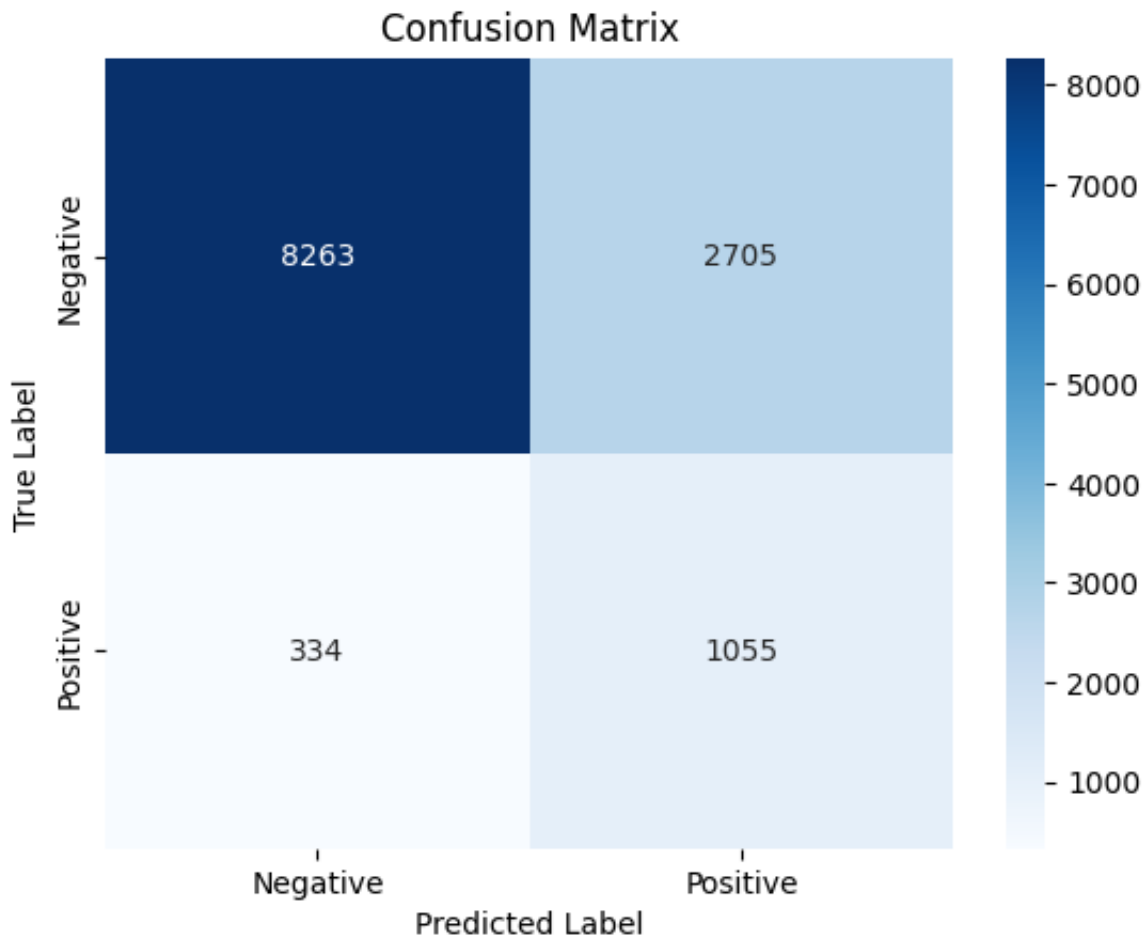


1b.) Confusion matrix on out of sample data. Visualize and store as variable

In [63]:
```
y_pred = dtree.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

In [64]:
```
class_labels = ['Negative', 'Positive']

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



3.) Use bagging on your descision tree

In [65]:
```
dtree = DecisionTreeClassifier(max_depth = 3)
```

In [66]:
```
bagging = BaggingClassifier(estimator=dtree,
                            n_estimators = 100,
                            max_samples = 0.5,
                            max_features = 1.)

bagging.fit(X_scaled,y_train)
```

```
/Users/archer/anaconda3/lib/python3.11/site-packages/sklearn/ensemble/_baggi
ng.py:782: DataConversionWarning: A column-vector y was passed when a 1d arr
ay was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
```

Out[66]:

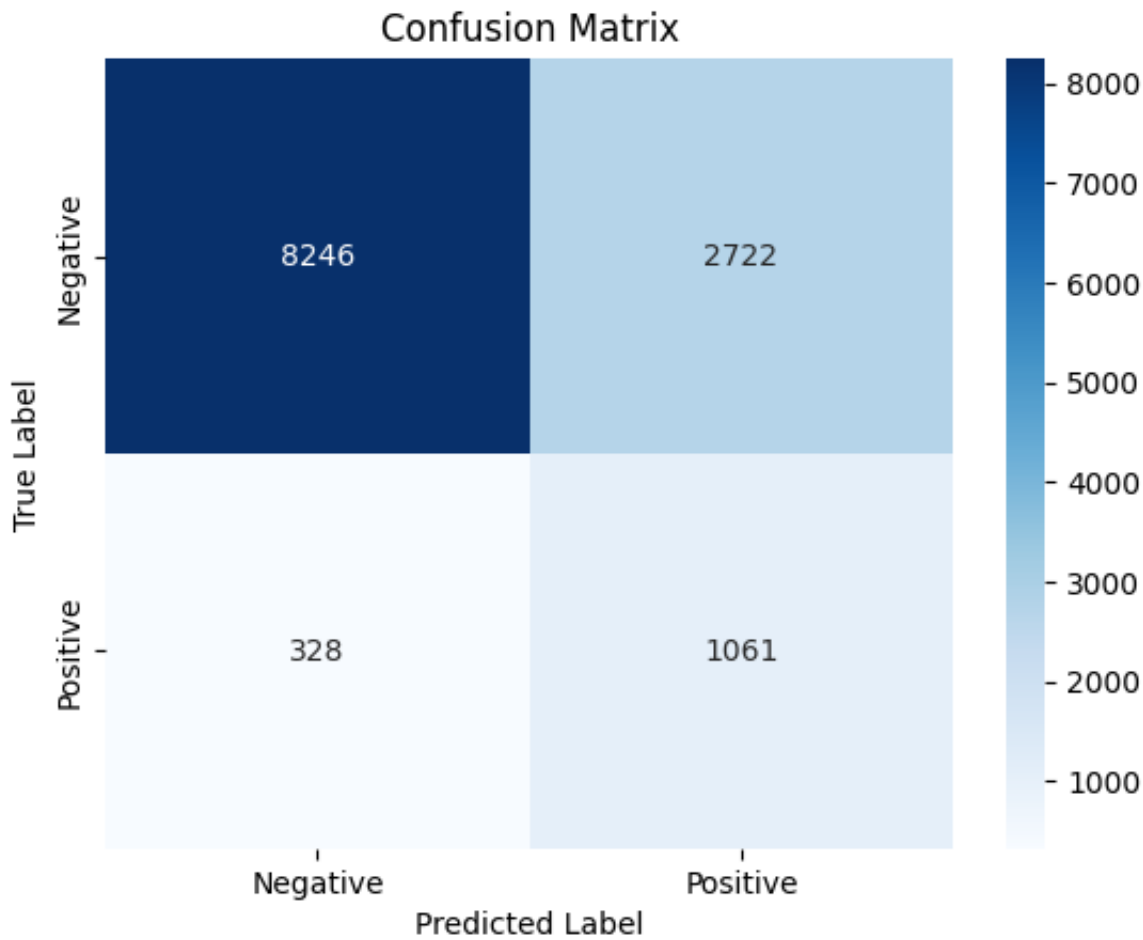
```
▸ BaggingClassifier
▸ estimator: DecisionTreeClassifier
  ▸ DecisionTreeClassifier
```

In [67]:

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

In [68]:

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



4.) Boost your tree

In [69]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [70]:
```
boost = AdaBoostClassifier(estimator=dtree, n_estimators = 100)
boost.fit(X_scaled,y_train)
```

```
/Users/archer/anaconda3/lib/python3.11/site-packages/sklearn/utils/validatio
n.py:1229: DataConversionWarning: A column-vector y was passed when a 1d arr
ay was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
/Users/archer/anaconda3/lib/python3.11/site-packages/sklearn/ensemble/_weigh
t_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is dep
recated and will be removed in 1.6. Use the SAMME algorithm to circumvent th
is warning.
  warnings.warn(
```

Out[70]:

```
▸ AdaBoostClassifier
▸ estimator: DecisionTreeClassifier
  ▸ DecisionTreeClassifier
```

In [71]:

```
AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=3),n_estimator
```

Out[71]:

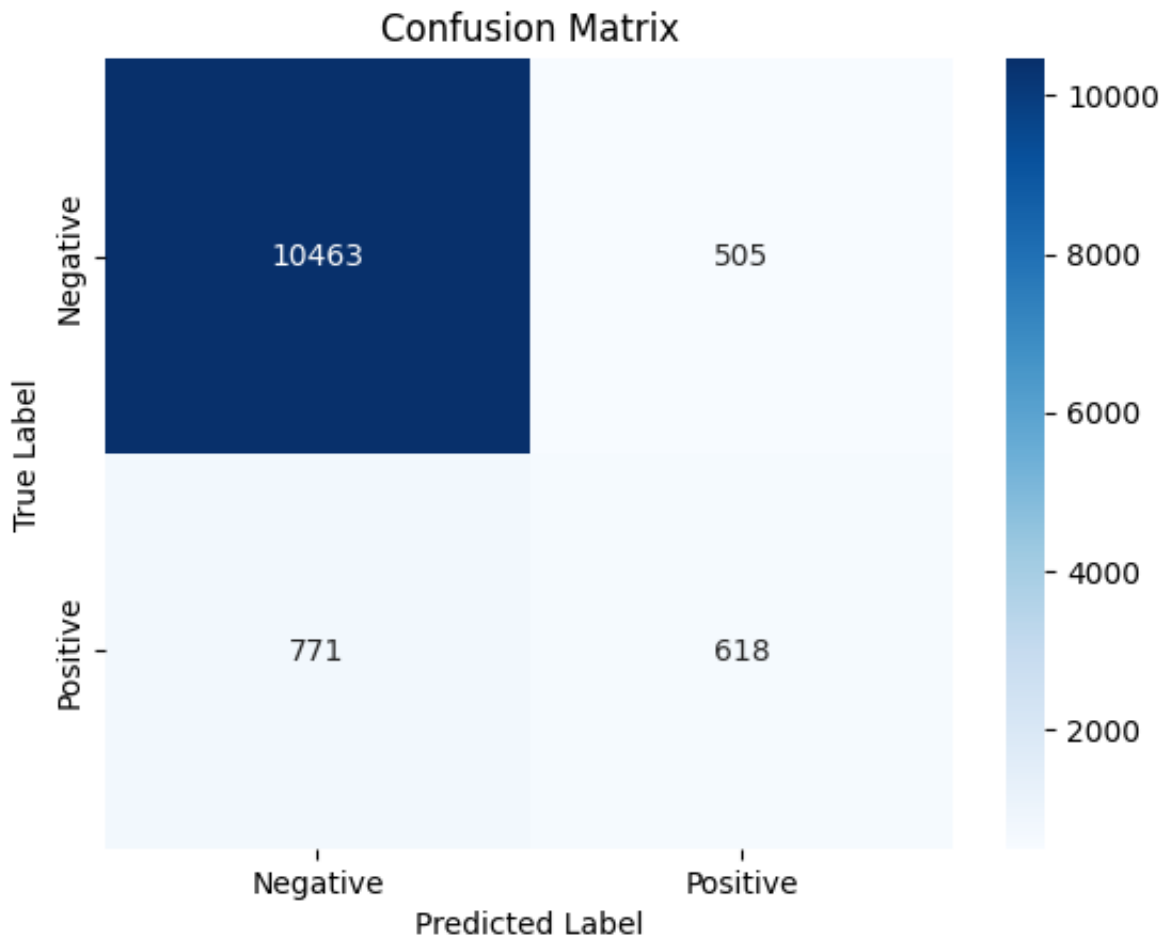
```
▸ AdaBoostClassifier
▸ estimator: DecisionTreeClassifier
  ▸ DecisionTreeClassifier
```

In [72]:

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

In [73]:

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



In [ ]:

5.) Create a superlearner with at least 4 base learner models. Use a logistic reg for your metalearner. Interpret your coefficients and save your CM.

In [57]:
```
pip install mlens
```

```
Requirement already satisfied: mlens in /Users/archer/anaconda3/lib/python3.
11/site-packages (0.2.3)
Requirement already satisfied: scipy>=0.17 in /Users/archer/anaconda3/lib/py
thon3.11/site-packages (from mlens) (1.10.1)
Requirement already satisfied: numpy>=1.11 in /Users/archer/anaconda3/lib/py
thon3.11/site-packages (from mlens) (1.24.3)
WARNING: You are using pip version 21.3.1; however, version 24.0 is availabl
e.
You should consider upgrading via the '/Users/archer/anaconda3/bin/python -m
pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

In [74]:
```
base_predictions =[list(dtree_main.predict(X_scaled)),
                   list(boost.predict(X_scaled)),
                   list(bagging.predict(X_scaled))]
```

In [75]:
```
n = len(base_predictions[0])
n
```

Out[75]:
```
51160
```

In [76]:
```
base_predictions = np.array(base_predictions).transpose()
```

In [77]:
```
super_learner = LogisticRegression()
```

In [78]:
```
super_learner.fit(base_predictions,y_train)
```

```
/Users/archer/anaconda3/lib/python3.11/site-packages/sklearn/utils/validatio
n.py:1229: DataConversionWarning: A column-vector y was passed when a 1d arr
ay was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
```

Out[78]:
```
▾ LogisticRegression
LogisticRegression()
```

In [79]:
```
LogisticRegression()
```

Out[79]:
```
▾ LogisticRegression
LogisticRegression()
```

In [80]:
```
super_learner.coef_
```

Out[80]:
```
array([[0.76004439, 5.32120717, 0.75141672]])
```

In [ ]:

In [ ]:

6.)

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: