Sec1_HW8

February 28, 2024

1 0.) Import and Clean data

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

[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
```

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

[4]:
```
df = pd.read_csv('bank-additional-full (1).csv', delimiter = ';')
```

[5]:
```
df.head()
```

[5]:
```
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
3   40     admin.  married     basic.6y       no      no   no  telephone
4   56   services  married  high.school       no      no  yes  telephone

  month day_of_week  …  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  …         1    999         0  nonexistent          1.1
1   may         mon  …         1    999         0  nonexistent          1.1
2   may         mon  …         1    999         0  nonexistent          1.1
3   may         mon  …         1    999         0  nonexistent          1.1
4   may         mon  …         1    999         0  nonexistent          1.1
```

```
   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

[6]:
```
df = df.drop(["default",␣
↪"pdays",        "previous",        "poutcome",        "emp.var.
↪rate",        "cons.price.idx",        "cons.conf.
↪idx",        "euribor3m",        "nr.employed"], axis = 1)
df = pd.get_dummies(df, columns = ["loan",␣
↪"job","marital","housing","contact","day_of_week", "campaign", "month",␣
↪"education"],drop_first = True)
```

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

[7]:
```
   age  duration   y  loan_unknown  loan_yes  job_blue-collar  \
0   56       261  no         False     False            False
1   57       149  no         False     False            False
2   37       226  no         False     False            False
3   40       151  no         False     False            False
4   56       307  no         False      True            False

   job_entrepreneur  job_housemaid  job_management  job_retired  …  \
0             False           True           False        False  …
1             False          False           False        False  …
2             False          False           False        False  …
3             False          False           False        False  …
4             False          False           False        False  …

   month_nov  month_oct  month_sep  education_basic.6y  education_basic.9y  \
0      False      False      False               False               False
1      False      False      False               False               False
2      False      False      False               False               False
3      False      False      False                True               False
4      False      False      False               False               False

   education_high.school  education_illiterate  education_professional.course  \
0                  False                 False                          False
1                   True                 False                          False
2                   True                 False                          False
3                  False                 False                          False
4                   True                 False                          False
```

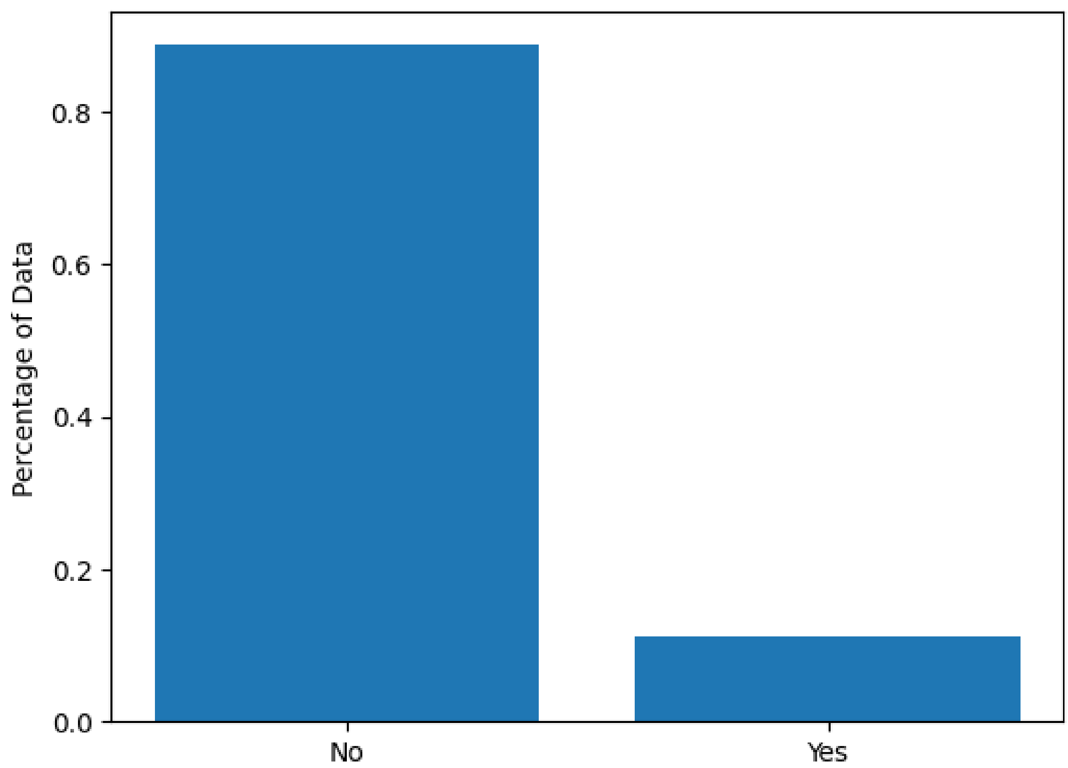
```
   education_university.degree  education_unknown
0                        False              False
1                        False              False
2                        False              False
3                        False              False
4                        False              False

[5 rows x 83 columns]
```

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

[ ]:
```
```

[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()
```



[10]:
```
# 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

[48]:
```
###############
###TRANSFORM###
###############
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state = 333)
smote_X, smote_y = smote.fit_resample(X_scaled, y_train)

X_scaled = smote_X
y_train = smote_y
```

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

[ ]:
```
```

[65]:
```
dtree = DecisionTreeClassifier(max_depth = 3)
dtree.fit(X_scaled, y_train)
```

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

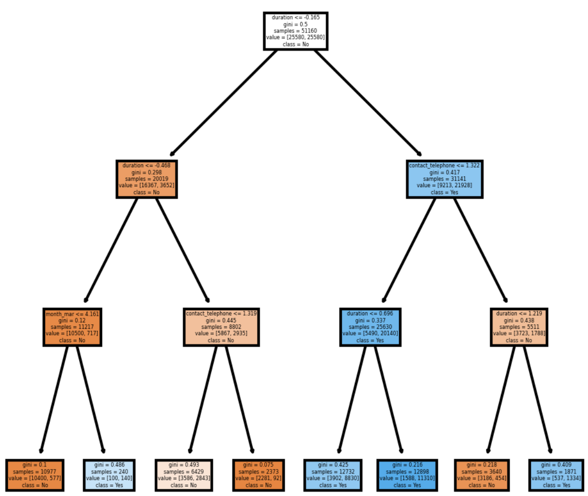
[50]:
```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(dtree, filled = True, feature_names = X.columns,␣
 ↪class_names=["No","Yes"])



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

[50]:
```
[Text(0.5, 0.875, 'duration <= -0.165\ngini = 0.5\nsamples = 51160\nvalue =
[25580, 25580]\nclass = No'),
 Text(0.25, 0.625, 'duration <= -0.468\ngini = 0.298\nsamples = 20019\nvalue =
[16367, 3652]\nclass = No'),
 Text(0.125, 0.375, 'month_mar <= 4.161\ngini = 0.12\nsamples = 11217\nvalue =
[10500, 717]\nclass = No'),
 Text(0.0625, 0.125, 'gini = 0.1\nsamples = 10977\nvalue = [10400, 577]\nclass =
No'),
 Text(0.1875, 0.125, 'gini = 0.486\nsamples = 240\nvalue = [100, 140]\nclass =
Yes'),
```

```
 Text(0.375, 0.375, 'contact_telephone <= 1.319\ngini = 0.445\nsamples =
8802\nvalue = [5867, 2935]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.493\nsamples = 6429\nvalue = [3586, 2843]\nclass
= No'),
 Text(0.4375, 0.125, 'gini = 0.075\nsamples = 2373\nvalue = [2281, 92]\nclass =
No'),
 Text(0.75, 0.625, 'contact_telephone <= 1.322\ngini = 0.417\nsamples =
31141\nvalue = [9213, 21928]\nclass = Yes'),
 Text(0.625, 0.375, 'duration <= 0.696\ngini = 0.337\nsamples = 25630\nvalue =
[5490, 20140]\nclass = Yes'),
 Text(0.5625, 0.125, 'gini = 0.425\nsamples = 12732\nvalue = [3902, 8830]\nclass
= Yes'),
 Text(0.6875, 0.125, 'gini = 0.216\nsamples = 12898\nvalue = [1588,
11310]\nclass = Yes'),
 Text(0.875, 0.375, 'duration <= 1.219\ngini = 0.438\nsamples = 5511\nvalue =
[3723, 1788]\nclass = No'),
 Text(0.8125, 0.125, 'gini = 0.218\nsamples = 3640\nvalue = [3186, 454]\nclass =
No'),
 Text(0.9375, 0.125, 'gini = 0.409\nsamples = 1871\nvalue = [537, 1334]\nclass =
Yes')]
```

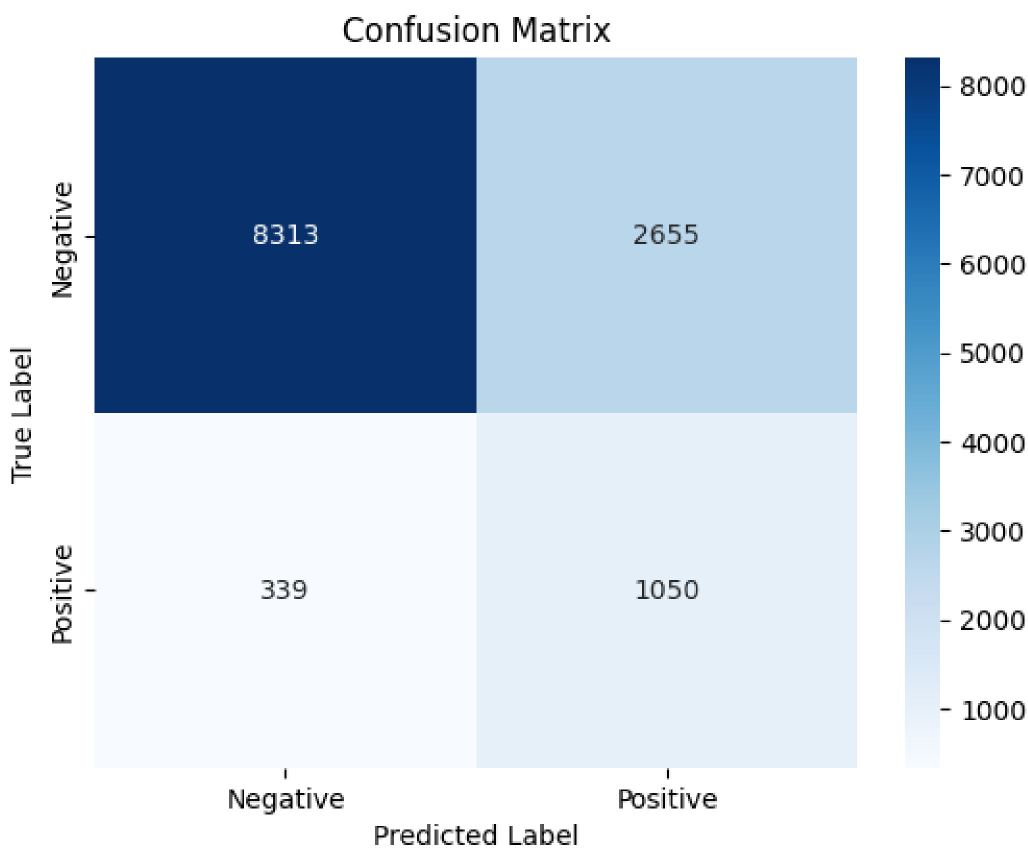


3 1b.) Confusion matrix on out of sample data. Visualize and store as variable

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

[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()
```



4 3.) Use bagging on your descision tree

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

[54]:
```
bagging = BaggingClassifier(estimator = dtree,
                            n_estimators = 100, # to optimize
                            max_samples = 0.5, # to optimize
                            max_features = 1.)

bagging.fit(X_scaled, y_train.yes)
```

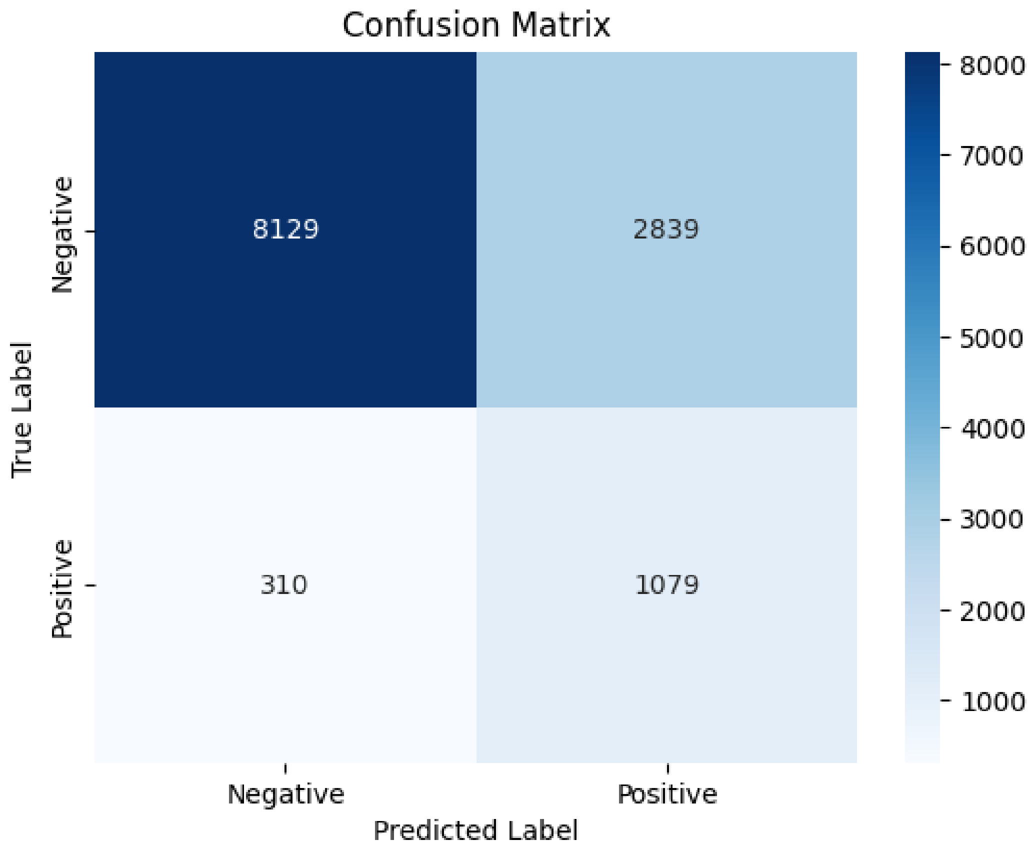
[54]:
```
BaggingClassifier(estimator=DecisionTreeClassifier(max_depth=3),
                  max_samples=0.5, n_estimators=100)
```

[55]:
```
y_pred = bagging.predict(X_test)
```

[56]:
```
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

[57]:
```
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 4.) Boost your tree

[58]:
```python
from sklearn.ensemble import AdaBoostClassifier
```

[59]:
```python
boost = AdaBoostClassifier(estimator = dtree,
                           n_estimators = 100) # to optimize

boost.fit(X_scaled, y_train.yes)
```

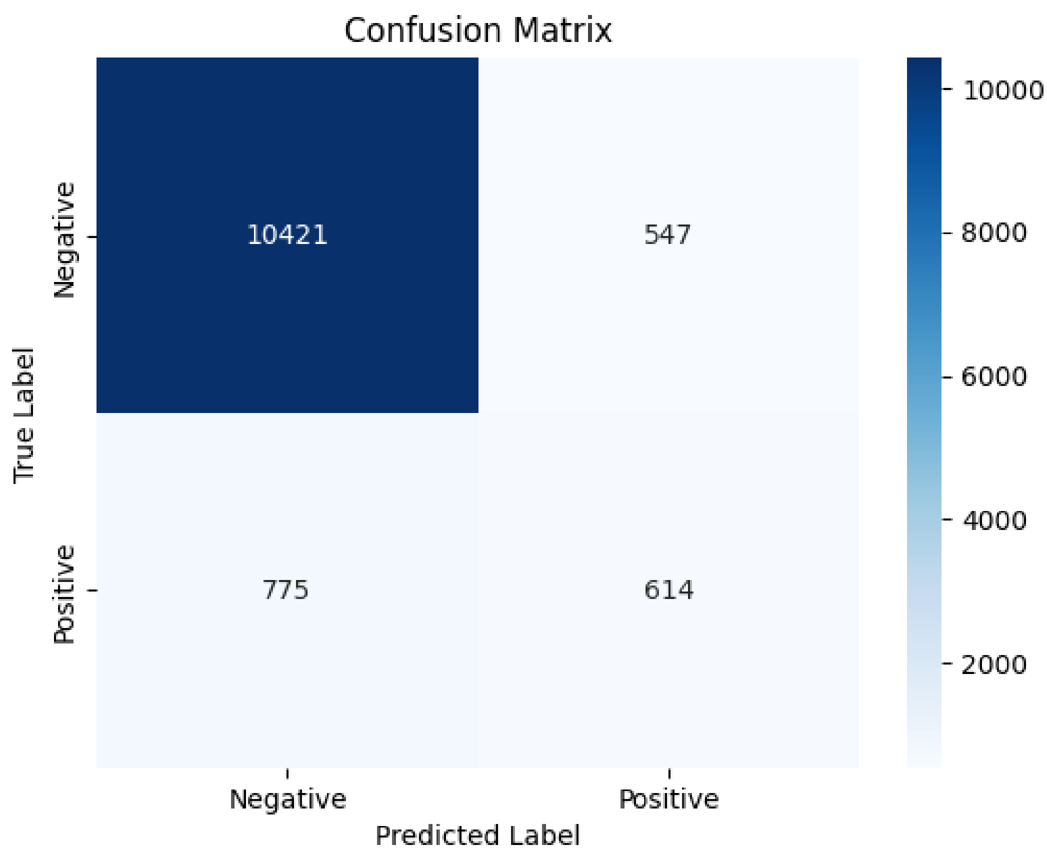
[59]:
```
AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=3),
                   n_estimators=100)
```

[60]:
```python
y_pred = boost.predict(X_test)
```

[61]:
```python
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

[62]:
```python
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()
```



6 5.) Train a logistic regression on the decision tree, boosted tree and bagegd tree. Interpret coefficients and significance

[66]:
```python
data = {
  "Intercept": np.ones(len(y_train)),
  "Bagging": bagging.predict(X_scaled),
  "Boosting": boost.predict(X_scaled),
  "Tree": dtree.predict(X_scaled),
```

```
}

x_base = pd.DataFrame(data)
x_base.head()
```

[66]:

	Intercept	Bagging	Boosting	Tree
0	1.0	False	False	False
1	1.0	False	False	False
2	1.0	False	False	False
3	1.0	False	False	False
4	1.0	True	False	True

[67]:
```
super_learner = LogisticRegression()
```

[68]:
```
super_learner.fit(x_base,y_train.yes)
```

[68]: LogisticRegression()

[69]:
```
# Get feature names
feature_names = x_base.columns  # Replace this with your actual feature names

# Print coefficients along with feature names
for feature_name, coef in zip(feature_names, super_learner.coef_[0]):
    print(feature_name + ":", coef)
```

```
Intercept: -0.0003262928713537420З
Bagging: 1.0853119198308605
Boosting: 5.341499509196074
Tree: 0.43208029230619616
```

[70]:
```
import statsmodels.api as sm

logit_model = sm.Logit(y_train.yes.astype(int), x_base.astype(int))
result = logit_model.fit()

print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.188663
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                    yes   No. Observations:                51160
Model:                          Logit   Df Residuals:                    51156
Method:                           MLE   Df Model:                            3
Date:                Wed, 28 Feb 2024   Pseudo R-squ.:                  0.7278
Time:                        15:58:50   Log-Likelihood:                -9652.0
converged:                       True   LL-Null:                       -35461.
Covariance Type:            nonrobust   LLR p-value:                     0.000
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -3.2908      0.035    -94.822      0.000      -3.359      -3.223
Bagging        1.0959      0.126      8.699      0.000       0.849       1.343
Boosting       5.3510      0.042    125.930      0.000       5.268       5.434
Tree           0.4220      0.125      3.381      0.001       0.177       0.667
==============================================================================
```

All three models seems to have a positive and significant effect on predicting the variable. This indicates that all models have information regarding the predicted series. The relative high magnitude on the boosting coefficient indicates that Boosting contributes the most among the tree in this super learner model. Looking at the confusion matrix, we are actually just repeating the predictions made by boosting.

[42]:
```
data = {
  "Intercept": np.ones(len(y_test)),
  "Bagging": bagging.predict(X_test),
  "Boosting": boost.predict(X_test),
  "Tree": dtree.predict(X_test),
}

x_test = pd.DataFrame(data)
x_test.head()
```

[42]:

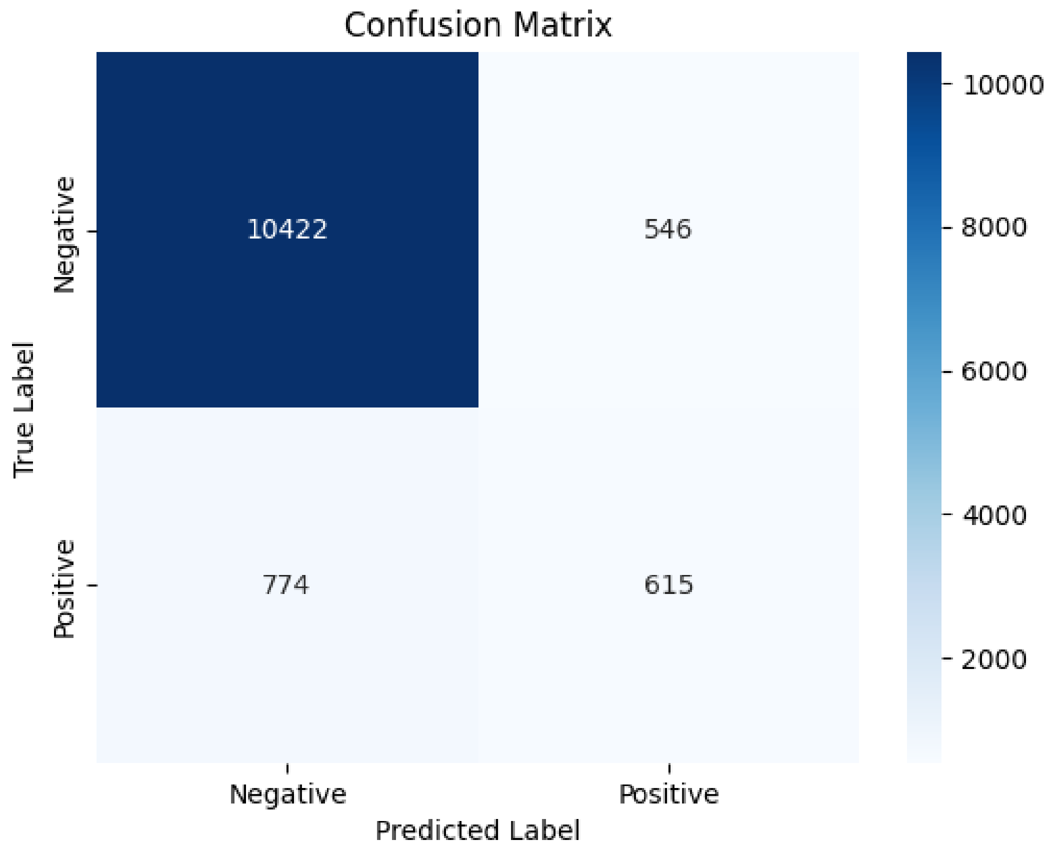
	Intercept	Bagging	Boosting	Tree
0	1.0	True	False	True
1	1.0	False	False	False
2	1.0	False	False	False
3	1.0	False	False	False
4	1.0	False	False	False

[43]:
```
y_pred = super_learner.predict(x_test)
```

[44]:
```
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

[45]:
```
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()
```



Confusion Matrix
10422
546
774
615
Negative
Positive
True Label
Negative
Positive
Predicted Label
10000
8000
6000
4000
2000