Sec1_HW8

February 29, 2024

1 0.) Import and Clean data

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

[ ]:
```
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 imblearn.over_sampling import KMeansSMOTE
```

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

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

[ ]:
```
df
```

[ ]:

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

```
         contact month day_of_week  …  campaign  pdays  previous  \
0      telephone   may         mon  …         1    999         0
1      telephone   may         mon  …         1    999         0
2      telephone   may         mon  …         1    999         0
3      telephone   may         mon  …         1    999         0
4      telephone   may         mon  …         1    999         0
…            …   …         …  …       …    …       …
41183   cellular   nov         fri  …         1    999         0
41184   cellular   nov         fri  …         1    999         0
41185   cellular   nov         fri  …         2    999         0
41186   cellular   nov         fri  …         1    999         0
41187   cellular   nov         fri  …         3    999         1

          poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0      nonexistent          1.1          93.994          -36.4      4.857
1      nonexistent          1.1          93.994          -36.4      4.857
2      nonexistent          1.1          93.994          -36.4      4.857
3      nonexistent          1.1          93.994          -36.4      4.857
4      nonexistent          1.1          93.994          -36.4      4.857
…              …          …              …              …        …
41183  nonexistent         -1.1          94.767          -50.8      1.028
41184  nonexistent         -1.1          94.767          -50.8      1.028
41185  nonexistent         -1.1          94.767          -50.8      1.028
41186  nonexistent         -1.1          94.767          -50.8      1.028
41187      failure         -1.1          94.767          -50.8      1.028

       nr.employed    y
0           5191.0   no
1           5191.0   no
2           5191.0   no
3           5191.0   no
4           5191.0   no
…              …  …
41183       4963.6  yes
41184       4963.6   no
41185       4963.6   no
41186       4963.6  yes
41187       4963.6   no

[41188 rows x 21 columns]
```

[ ]:
```
df = df.drop(["default",␣
 ↪"pdays",        "previous",        "poutcome",        "emp.var.
 ↪rate",        "cons.price.idx",        "cons.conf.
 ↪idx",        "euribor3m",        "nr.employed"], axis = 1)
```

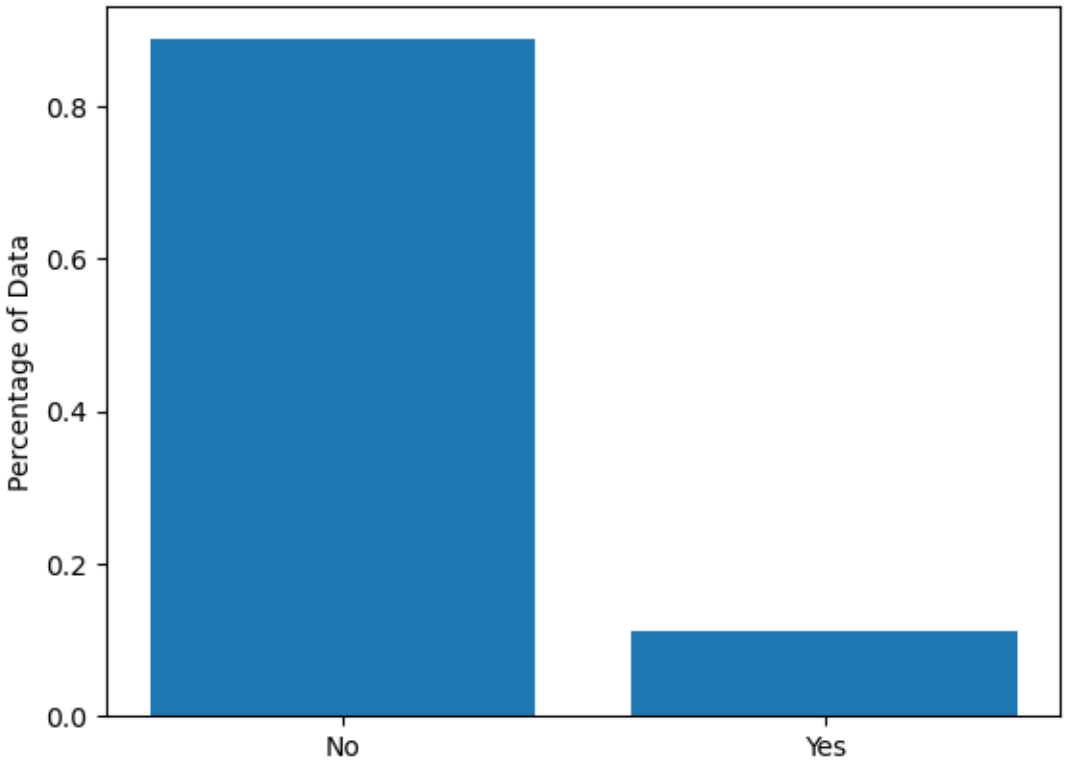
```
df = pd.get_dummies(df, columns = ["loan",␣
↪"job","marital","housing","contact","day_of_week", "campaign", "month",␣
↪"education"],drop_first = True)
```

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

[ ]:
```
   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]
```

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

[ ]:
```
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()
```



[ ]:
```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X.astype(int), y.
↪astype(int), test_size=0.3, random_state=42, )
```

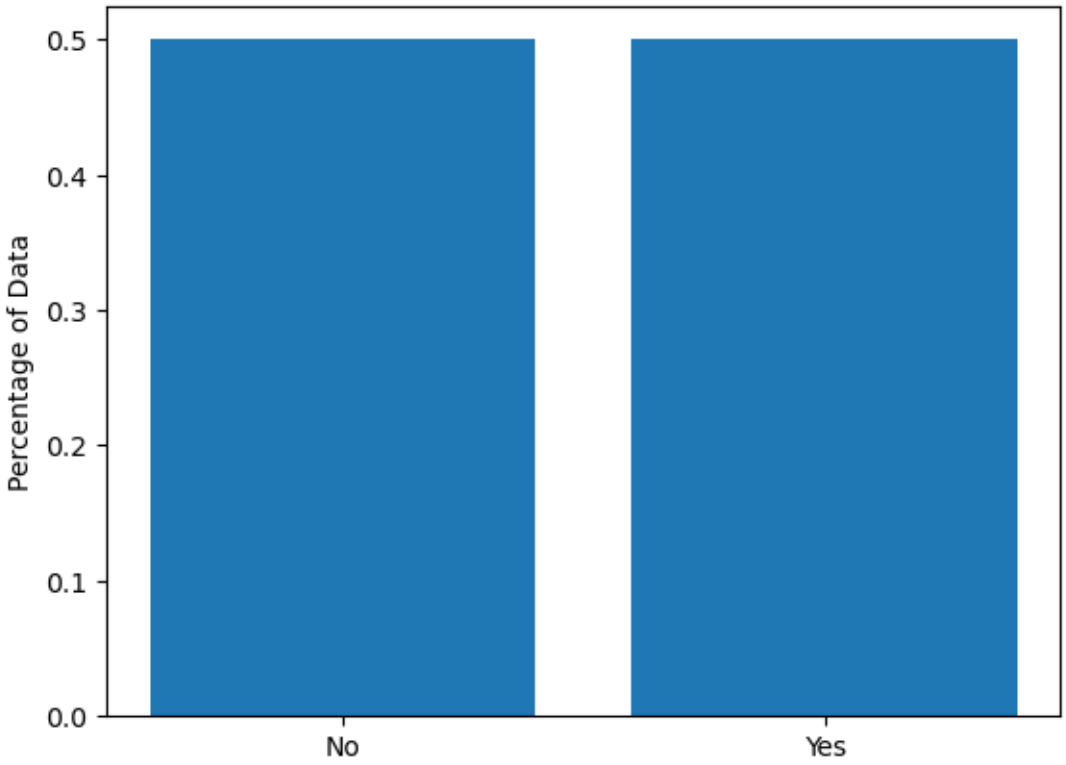
#1.) Based on the visualization above, use your expert opinion to transform the data based on what we learned this quarter

[ ]:
```
# We use the SMOTE algorithm to balance the data set

smote = KMeansSMOTE(random_state = 42)

X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

```
c:\Users\nikpa\anacondafinal\Lib\site-packages\sklearn\cluster\_kmeans.py:1930:
FutureWarning: The default value of `n_init` will change from 3 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=3)
c:\Users\nikpa\anacondafinal\Lib\site-packages\sklearn\cluster\_kmeans.py:1962:
```

```
UserWarning: MiniBatchKMeans is known to have a memory leak on Windows with MKL,
when there are less chunks than available threads. You can prevent it by setting
batch_size >= 3072 or by setting the environment variable OMP_NUM_THREADS=4
  warnings.warn(
```

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



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

[ ]:
```
dtree_main = DecisionTreeClassifier(max_depth = 3)
dtree_main.fit(X_train_smote, y_train_smote)
```

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

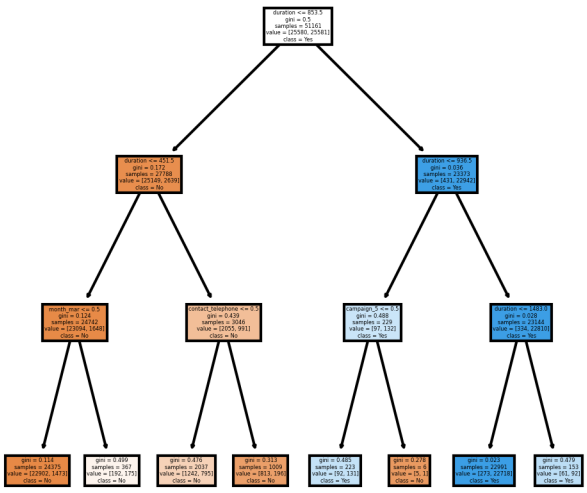
[ ]:
```
X_train_smote.head()
```

[ ]:
```
   age  duration  loan_unknown  loan_yes  job_blue-collar  job_entrepreneur  \
0   29        77             0         0                0                 0
1   29        12             0         0                0                 0
2   45       277             0         0                1                 0
3   34        70             0         0                0                 0
4   32      1181             0         0                0                 0

   job_housemaid  job_management  job_retired  job_self-employed  …  \
0              0               0            0                  0  …
1              0               0            0                  0  …
2              0               0            0                  0  …
3              0               0            0                  0  …
4              0               0            0                  0  …

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

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

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

[5 rows x 82 columns]
```

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

[ ]:
```
[Text(0.5, 0.875, 'duration <= 853.5\ngini = 0.5\nsamples = 51161\nvalue =
[25580, 25581]\nclass = Yes'),
 Text(0.25, 0.625, 'duration <= 451.5\ngini = 0.172\nsamples = 27788\nvalue =
[25149, 2639]\nclass = No'),
 Text(0.125, 0.375, 'month_mar <= 0.5\ngini = 0.124\nsamples = 24742\nvalue =
[23094, 1648]\nclass = No'),
```

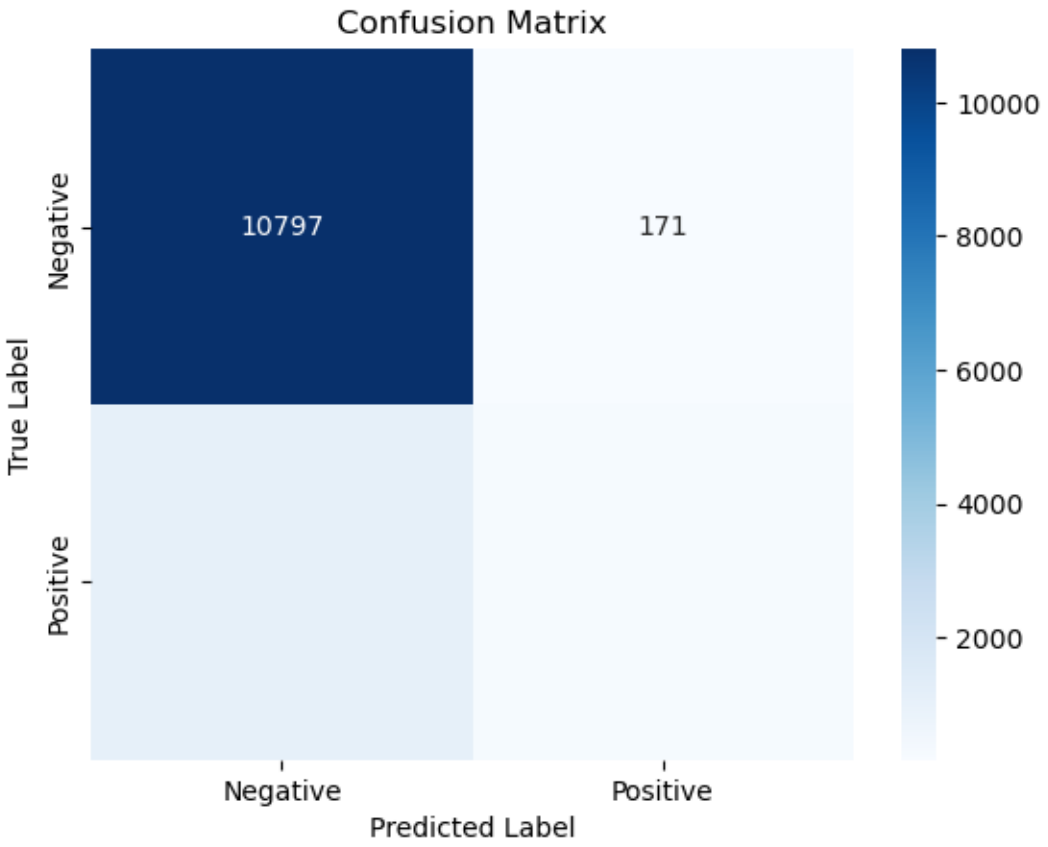
```
 Text(0.0625, 0.125, 'gini = 0.114\nsamples = 24375\nvalue = [22902,
1473]\nclass = No'),
 Text(0.1875, 0.125, 'gini = 0.499\nsamples = 367\nvalue = [192, 175]\nclass =
No'),
 Text(0.375, 0.375, 'contact_telephone <= 0.5\ngini = 0.439\nsamples =
3046\nvalue = [2055, 991]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.476\nsamples = 2037\nvalue = [1242, 795]\nclass =
No'),
 Text(0.4375, 0.125, 'gini = 0.313\nsamples = 1009\nvalue = [813, 196]\nclass =
No'),
 Text(0.75, 0.625, 'duration <= 936.5\ngini = 0.036\nsamples = 23373\nvalue =
[431, 22942]\nclass = Yes'),
 Text(0.625, 0.375, 'campaign_5 <= 0.5\ngini = 0.488\nsamples = 229\nvalue =
[97, 132]\nclass = Yes'),
 Text(0.5625, 0.125, 'gini = 0.485\nsamples = 223\nvalue = [92, 131]\nclass =
Yes'),
 Text(0.6875, 0.125, 'gini = 0.278\nsamples = 6\nvalue = [5, 1]\nclass = No'),
 Text(0.875, 0.375, 'duration <= 1483.0\ngini = 0.028\nsamples = 23144\nvalue =
[334, 22810]\nclass = Yes'),
 Text(0.8125, 0.125, 'gini = 0.023\nsamples = 22991\nvalue = [273, 22718]\nclass
= Yes'),
 Text(0.9375, 0.125, 'gini = 0.479\nsamples = 153\nvalue = [61, 92]\nclass =
Yes')]
```



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

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

[ ]:
```
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

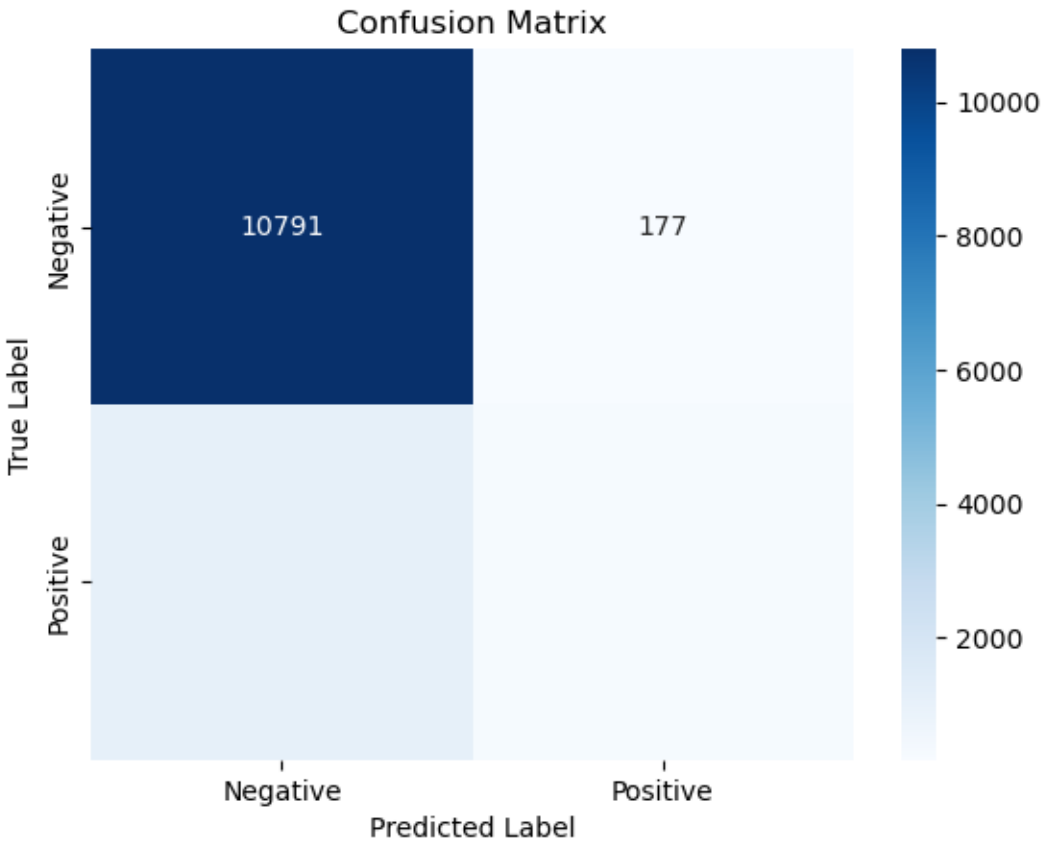
[ ]:
```
bagging_dtree = DecisionTreeClassifier(max_depth=3)
```

[ ]:
```
bagging= BaggingClassifier(estimator=bagging_dtree,
                  n_estimators= 100,
                  max_samples= .5,
                  max_features=1.)

bagging.fit(X_train_smote, y_train_smote)

y_pred_bagging = bagging.predict
```

```
c:\Users\nikpa\anacondafinal\Lib\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)
```

[ ]:
```
y_pred_bagging = bagging.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred_bagging)
```

[ ]:
```
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

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

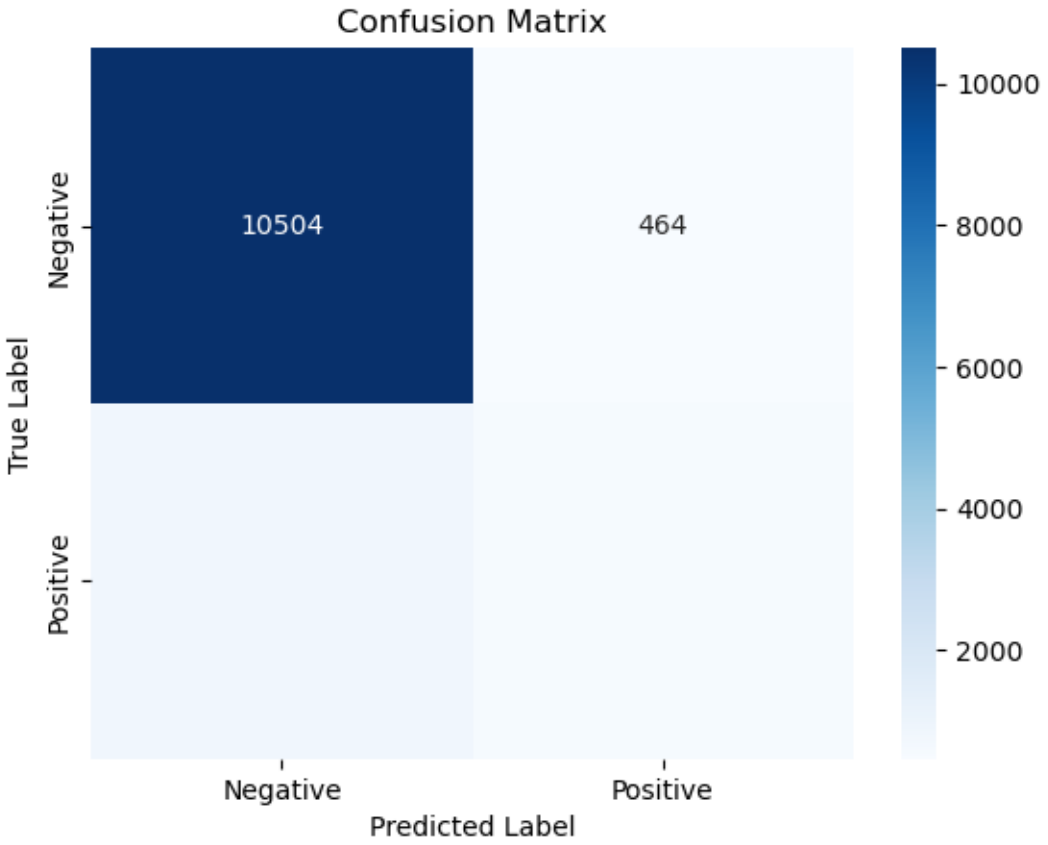
[ ]:
```
boost_dtree = DecisionTreeClassifier(max_depth=3)
```

[ ]:
```
boost = AdaBoostClassifier(estimator=boost_dtree,
                  n_estimators= 100,
                  )

boost.fit(X_train_smote, y_train_smote)

y_pred_boost = boost.predict
```

```
c:\Users\nikpa\anacondafinal\Lib\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)
```

[ ]:
```
y_pred_boost = boost.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred_boost)
```

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

Train a Logistic Rgression on the outputs

[ ]:
```
pip install mlens
```

```
Requirement already satisfied: mlens in c:\users\nikpa\anacondafinal\lib\site-
packages (0.2.3)
Requirement already satisfied: numpy>=1.11 in
c:\users\nikpa\anacondafinal\lib\site-packages (from mlens) (1.24.3)
Requirement already satisfied: scipy>=0.17 in
c:\users\nikpa\anacondafinal\lib\site-packages (from mlens) (1.11.4)
Note: you may need to restart the kernel to use updated packages.
```

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

[ ]:
```
bagging.predict(X_train_smote)
boost.predict(X_train_smote)
dtree_main.predict(X_train_smote)
```

[ ]:
```
array([0, 0, 0, …, 1, 1, 1])
```

[ ]:
```
# Put the in a list
X_base_learners = np.array(
    [
        (bagging.predict_proba(X_train_smote)[:,1]),
        boost.predict_proba(X_train_smote)[:,1],
        dtree_main.predict_proba(X_train_smote)[:,1],
     ]).T
```

[ ]:
```
X_base_learners = pd.DataFrame(X_base_learners)
```

[ ]:
```
supper_learner = LogisticRegression()
```

[ ]:
```
supper_learner.fit(X_base_learners, y_train_smote)
```

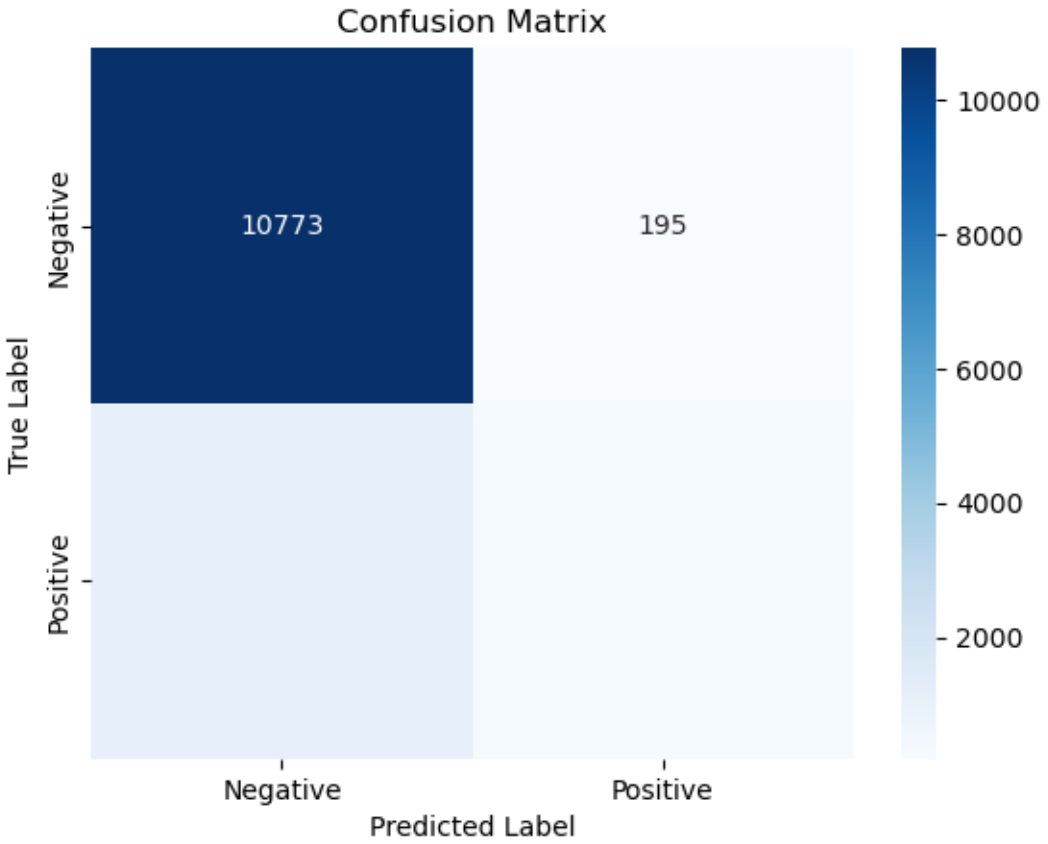
```
c:\Users\nikpa\anacondafinal\Lib\site-packages\sklearn\utils\validation.py:1184:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
# X test transfromed
X_test_learner = np.array(
    [
        (bagging.predict_proba(X_test)[:,1]),
        boost.predict_proba(X_test)[:,1],
        dtree_main.predict_proba(X_test)[:,1],
     ]).T
```

[ ]:
```
y_pred_super_learner = supper_learner.predict(X_test_learner)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred_super_learner)
```

[ ]:
```
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()
```



[ ]:
```
supper_learner.coef_
```

[ ]:
```
array([[ 7.24762   , 25.16308257, -0.72406826]])
```

The Boosting model appears to be the most influential among the three base learners, followed by the Bagging model. The Decision Tree model, while considered, has a relatively minor and negative impact on the superlearner's output. Overall thought, we don't observe any important change in the predictive power of the model, especially if we compare it with the boostng.