0.) Import and Clean data

```
import pandas as pd
from google.colab import drive
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
import warnings
warnings.filterwarnings('ignore')


#drive.mount('/content/gdrive/', force_remount = True)


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


df.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_o
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	57	services	married	high.school	unknown	no	no	telephone	may	
2	37	services	married	high.school	no	yes	no	telephone	may	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	
4	56	services	married	high.school	no	no	yes	telephone	may	

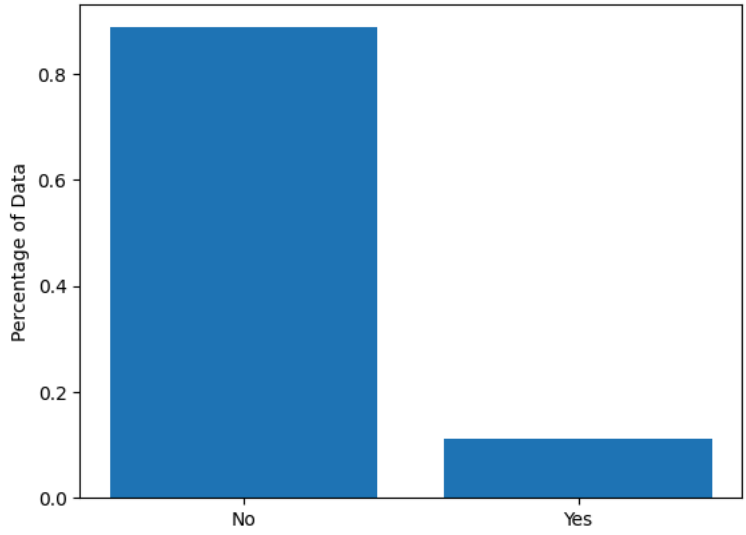
5 rows × 21 columns

```
df = df.drop(["default", "pdays",  "previous", "poutcome", "emp.var.rate", "cons.price.idx",  "cons.conf.idx",  "euribor3m",  "nr
df = pd.get_dummies(df, columns = ["loan", "job","marital","housing","contact","day_of_week", "campaign", "month", "education"],drop_fi


df.head()
```

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

5 rows × 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, 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

```
###############
###TRANSFORM###
###############

#X_scaled = #???
#y_train = #???
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_scaled, y_train = smote.fit_resample(X_scaled, y_train)
```

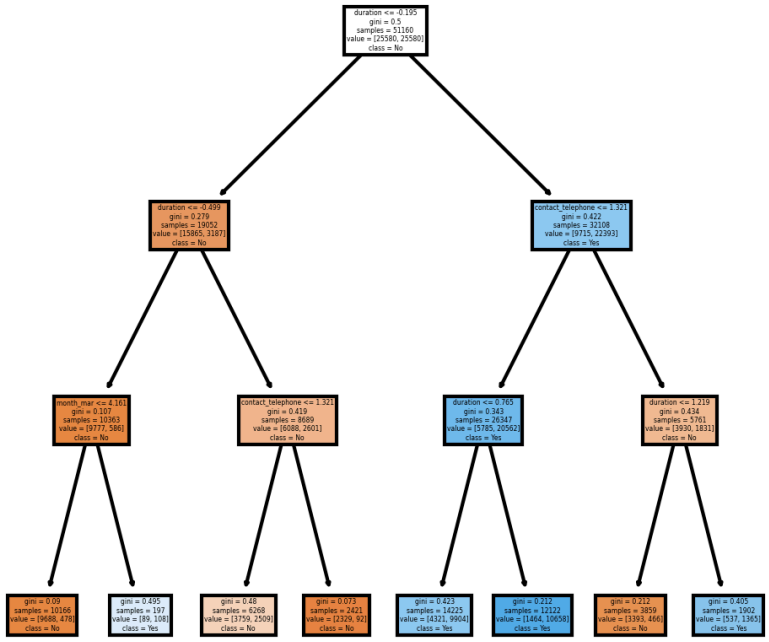
2.) Build and visualize a decision tree of Max Depth 3. Show the confusion matrix.

```
dtree1 = DecisionTreeClassifier(max_depth = 3)
dtree1.fit(X_scaled, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(dtree1, filled = True, feature_names = X.columns, class_names=["No","Yes"])


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

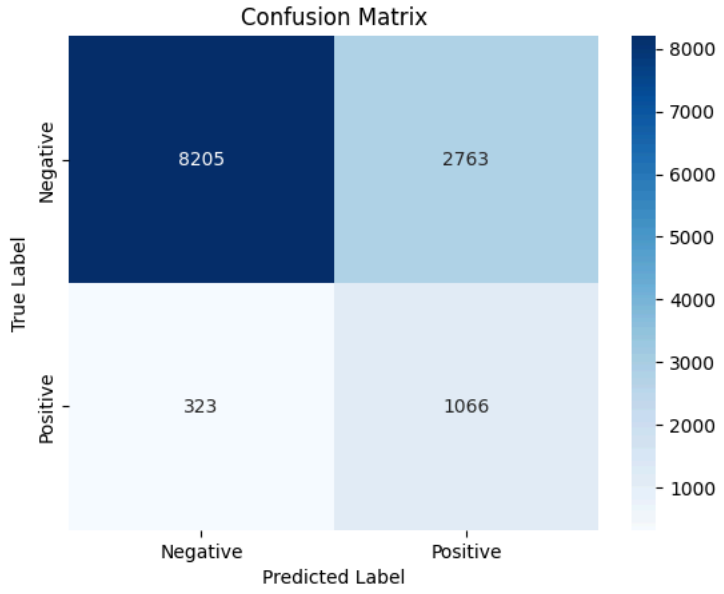
```
Yes'),
 Text(0.375, 0.375, 'contact_telephone <= 1.321\ngini = 0.419\nsamples =
8689\nvalue = [6088, 2601]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.48\nsamples = 6268\nvalue = [3759, 2509]\nclass =
No'),
 Text(0.4375, 0.125, 'gini = 0.073\nsamples = 2421\nvalue = [2329, 92]\nclass =
No'),
 Text(0.75, 0.625, 'contact_telephone <= 1.321\ngini = 0.422\nsamples =
32108\nvalue = [9715, 22393]\nclass = Yes'),
 Text(0.625, 0.375, 'duration <= 0.765\ngini = 0.343\nsamples = 26347\nvalue =
[5785, 20562]\nclass = Yes'),
 Text(0.5625, 0.125, 'gini = 0.423\nsamples = 14225\nvalue = [4321, 9904]\nclass
= Yes'),
 Text(0.6875, 0.125, 'gini = 0.212\nsamples = 12122\nvalue = [1464, 10658]\nclass
= Yes'),
 Text(0.875, 0.375, 'duration <= 1.219\ngini = 0.434\nsamples = 5761\nvalue =
[3930, 1831]\nclass = No'),
 Text(0.8125, 0.125, 'gini = 0.212\nsamples = 3859\nvalue = [3393, 466]\nclass =
No'),
 Text(0.9375, 0.125, 'gini = 0.405\nsamples = 1902\nvalue = [537, 1365]\nclass =
Yes')]
```



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

```
y_pred = dtree1.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

```
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

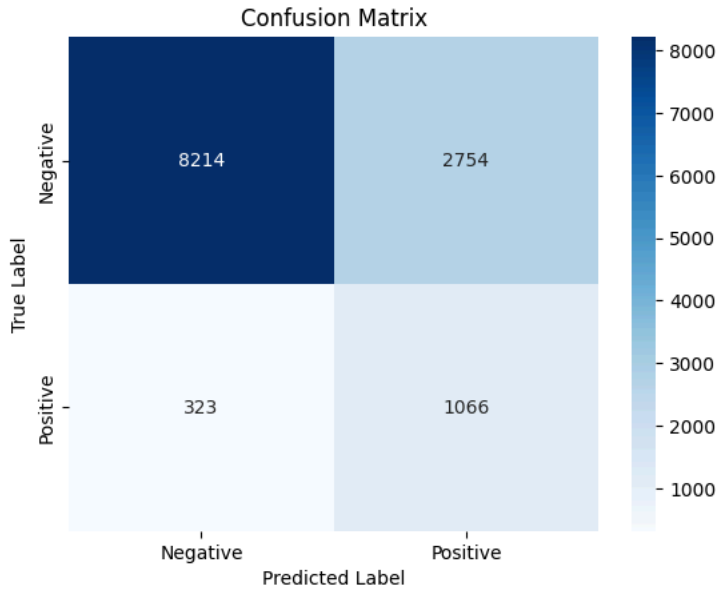
```
dtree=DecisionTreeClassifier(max_depth=3)
dtree.fit(X_scaled, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

```
bagging = BaggingClassifier(estimator = dtree,
                            n_estimators = 100,
                            max_samples = 0.5,
                            max_features = 1.)
bagging.fit(X_scaled, y_train)
y_pred = bagging.predict(X_test)
```

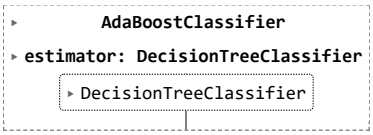
```
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

```
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()
```

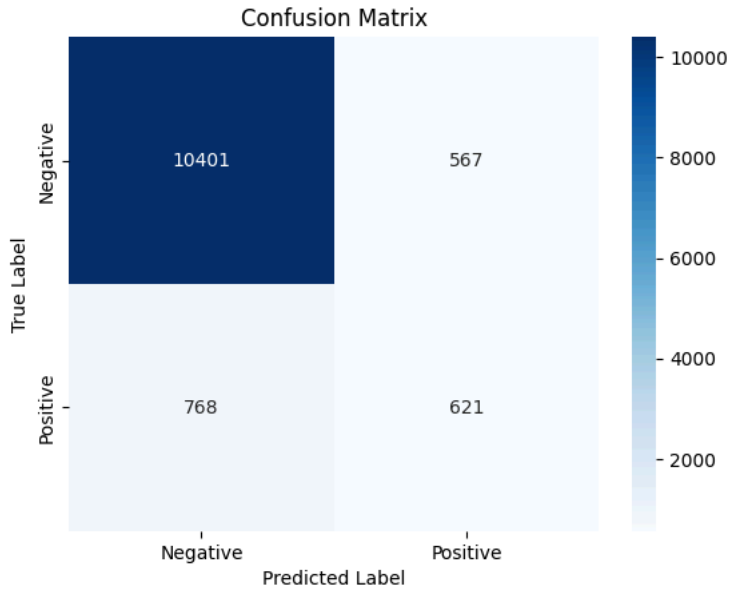


4.) Boost your tree

```
from sklearn.ensemble import AdaBoostClassifier


dtree = DecisionTreeClassifier(max_depth = 3)


boost = AdaBoostClassifier(estimator = dtree,
                           n_estimators = 50)
boost.fit(X_scaled, y_train)
```




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


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

```
pip install mlens
```

```
Collecting mlens
  Downloading mlens-0.2.3-py2.py3-none-any.whl (227 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 227.7/227.7 kB 2.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.10/dist-packages (from mlens) (1.25.2)
Requirement already satisfied: scipy>=0.17 in /usr/local/lib/python3.10/dist-packages (from mlens) (1.11.4)
Installing collected packages: mlens
Successfully installed mlens-0.2.3
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
X_base_learners = [list(bagging.predict(X_scaled)), list(boost.predict(X_scaled)), list(dtree.predict(X_scaled))]
```

```
superlearner=LogisticRegression()
```

```
X_base_learners = np.array(X_base_learners).transpose()
```

```
superlearner.fit(X_base_learners, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
print(superlearner.coef_)
```

```
[[1.26573401 4.95366648 0.18813718]]
```

the boosted tree model has the largest effects on prediction for its highest values of coefficients.

```
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator`
  warnings.warn(
```