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


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

```
Mounted at /content/gdrive/
```

```
df = pd.read_csv("/content/gdrive/MyDrive/ECON441B/bank-additional-full.csv", sep = ";")


df.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	

5 rows × 21 columns

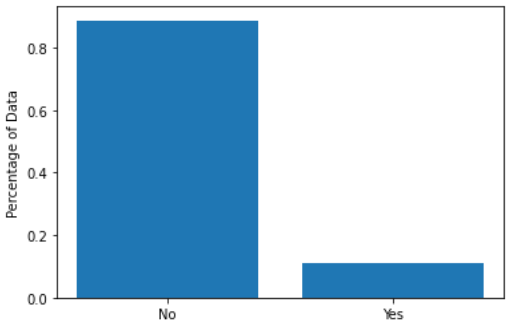
```
df = df.drop(["default", "pdays",   "previous", "poutcome", "emp.var.rate", "cons.price.idx",   "cons.conf.idx",    "euribor3m",    "nr.employe
df = pd.get_dummies(df, columns = ["loan", "job","marital","housing","contact","day_of_week", "campaign", "month", "education"],drop_first = Tr


df.head()
```

	age	duration	y	loan_unknown	loan_yes	job_blue-collar	job_entrepreneur	job_housemaid	job_management	job_retired	...	month_nov	month_
0	56	261	no	0	0	0	0	1	0	0	...	0	
1	57	149	no	0	0	0	0	0	0	0	...	0	
2	37	226	no	0	0	0	0	0	0	0	...	0	
3	40	151	no	0	0	0	0	0	0	0	...	0	
4	56	307	no	0	1	0	0	0	0	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###
###############
from imblearn.over_sampling import SMOTE
from sklearn.datasets import make_classification

smote = SMOTE(random_state = 42)
X_scaled, y_train = smote.fit_resample(X_scaled, y_train)
```

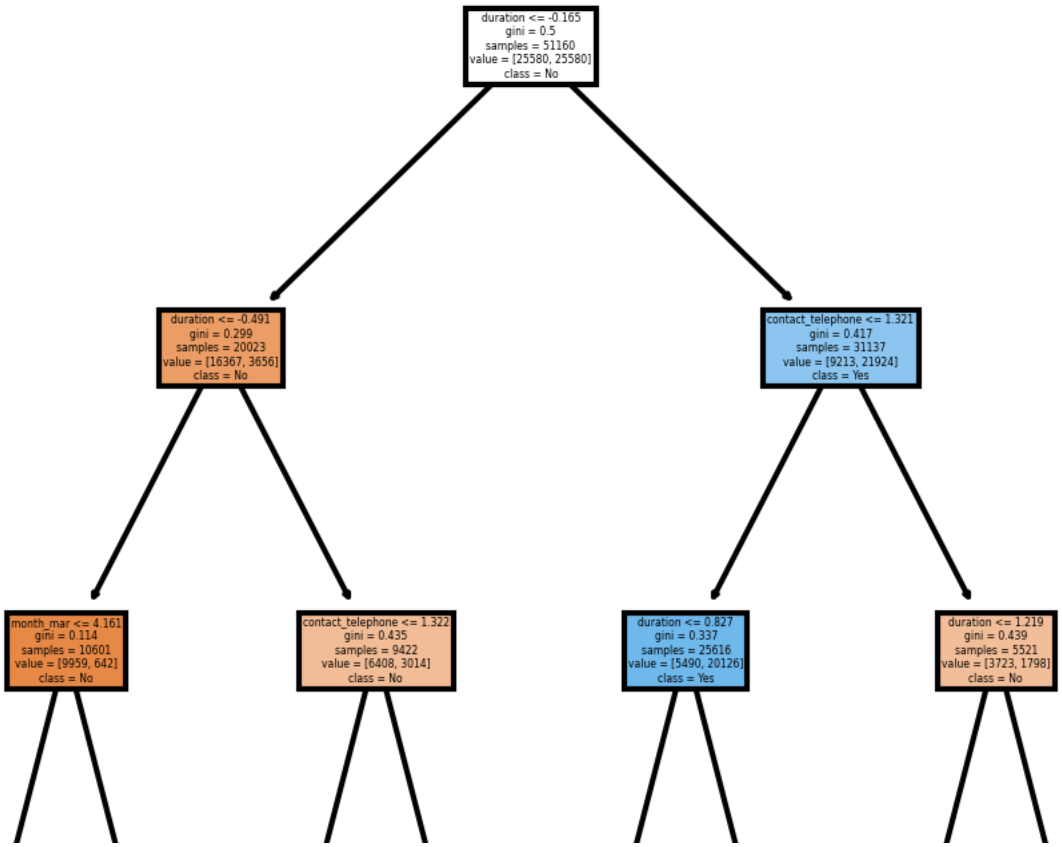
2.) Build and visualize a decision tree of Max Depth 3. Show the confusion matrix.

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

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

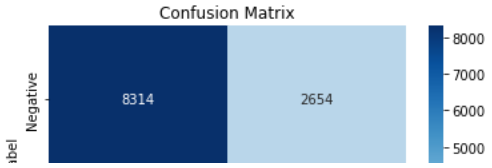
```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(dtree, filled = True, feature_names = X.columns, class_names=["No","Yes"])


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

```
[Text(0.5, 0.875, 'duration <= -0.165\ngini = 0.5\nsamples = 51160\nvalue = [25580, 25580]\nclass = No'),
 Text(0.25, 0.625, 'duration <= -0.491\ngini = 0.299\nsamples = 20023\nvalue = [16367, 3656]\nclass = No'),
 Text(0.125, 0.375, 'month_mar <= 4.161\ngini = 0.114\nsamples = 10601\nvalue = [9959, 642]\nclass = No'),
 Text(0.0625, 0.125, 'gini = 0.097\nsamples = 10397\nvalue = [9868, 529]\nclass = No'),
 Text(0.1875, 0.125, 'gini = 0.494\nsamples = 204\nvalue = [91, 113]\nclass = Yes'),
 Text(0.375, 0.375, 'contact_telephone <= 1.322\ngini = 0.435\nsamples = 9422\nvalue = [6408, 3014]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.489\nsamples = 6860\nvalue = [3934, 2926]\nclass = No'),
 Text(0.4375, 0.125, 'gini = 0.066\nsamples = 2562\nvalue = [2474, 88]\nclass = No'),
 Text(0.75, 0.625, 'contact_telephone <= 1.321\ngini = 0.417\nsamples = 31137\nvalue = [9213, 21924]\nclass = Yes'),
 Text(0.625, 0.375, 'duration <= 0.827\ngini = 0.337\nsamples = 25616\nvalue = [5490, 20126]\nclass = Yes'),
 Text(0.5625, 0.125, 'gini = 0.418\nsamples = 13908\nvalue = [4137, 9771]\nclass = Yes'),
 Text(0.6875, 0.125, 'gini = 0.204\nsamples = 11708\nvalue = [1353, 10355]\nclass = Yes'),
 Text(0.875, 0.375, 'duration <= 1.219\ngini = 0.439\nsamples = 5521\nvalue = [3723, 1798]\nclass = No'),
 Text(0.8125, 0.125, 'gini = 0.22\nsamples = 3645\nvalue = [3186, 459]\nclass = No'),
 Text(0.9375, 0.125, 'gini = 0.409\nsamples = 1876\nvalue = [537, 1339]\nclass = Yes')]
```



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


```
dty_pred = dtree.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, dty_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()
```

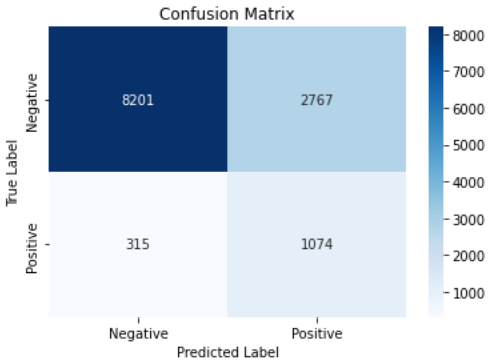


3.) Use bagging on your descision tree


```
dtree = DecisionTreeClassifier(max_depth = 3)

bagging = BaggingClassifier(estimator=dtree,
                            n_estimators=100,
                            max_samples=0.5,
                            max_features=1.)



bagging.fit(X_scaled, y_train)

bagy_pred = bagging.predict(X_test)

accuracy = accuracy_score(y_test, bagy_pred)
print("Accuracy:", accuracy)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_bagging.py:802: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
Accuracy: 0.7505867119851096
```

```
bagy_pred = bagging.predict(X_test)
y_true = y_test
cm_bag = confusion_matrix(y_true, bagy_pred)


class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_bag, 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()
```



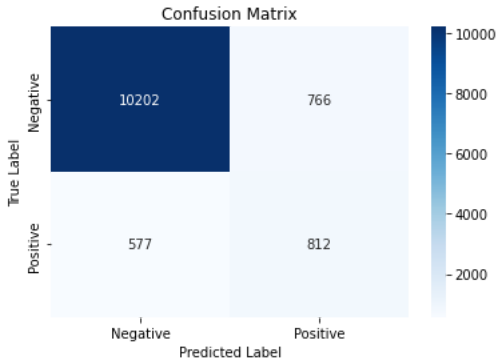
4.) Boost your tree

```
from sklearn.ensemble import AdaBoostClassifier


dtree = DecisionTreeClassifier(max_depth=3)
adaboost = AdaBoostClassifier(base_estimator=dtree, n_estimators=50, learning_rate=0.1)
```

```
adaboost.fit(X_scaled, y_train)

boosty_pred = adaboost.predict(X_test)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator` in versio
  warnings.warn(
```

```
boosty_pred = adaboost.predict(X_test)
y_true = y_test
cm_boost = confusion_matrix(y_true, boosty_pred)


class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_boost, 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 5 base learner models. Use a logistic reg for your metalearner. Interpret your coefficients and save your CM.

```
pip install mlens
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: mlens in /usr/local/lib/python3.8/dist-packages (0.2.3)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-packages (from mlens) (1.22.4)
Requirement already satisfied: scipy>=0.17 in /usr/local/lib/python3.8/dist-packages (from mlens) (1.10.1)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
####IMPORT MORE BASE LEARNERS####
from mlens.ensemble import SuperLearner


### SET YOUR BASE LEARNERS
base_estimators = [
    LogisticRegression(),
    RandomForestClassifier(),
    KNeighborsClassifier(n_neighbors = 5),
    AdaBoostClassifier(),
    DecisionTreeClassifier()
]
super_learner = SuperLearner()
super_learner.add(base_estimators)
### FIT TO TRAINING DATA
super_learner.fit(X_scaled, y_train)
### GET base_predictions
```

```
base_predictions = super_learner.predict(X_scaled)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/neighbors/_classification.py:215: DataConversionWarning: A column-vector y was passed when
  return self._fit(X, y)
/usr/local/lib/python3.8/dist-packages/sklearn/neighbors/_classification.py:215: DataConversionWarning: A column-vector y was passed when
  return self._fit(X, y)
/usr/local/lib/python3.8/dist-packages/sklearn/neighbors/_classification.py:215: DataConversionWarning: A column-vector y was passed when
  return self._fit(X, y)
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/mlens/parallel/learner.py:179: DataConversionWarning: A column-vector y was passed when a 1d array
  self.estimator.fit(xtemp, ytemp)
/usr/local/lib/python3.8/dist-packages/mlens/parallel/learner.py:179: DataConversionWarning: A column-vector y was passed when a 1d array
  self.estimator.fit(xtemp, ytemp)
/usr/local/lib/python3.8/dist-packages/mlens/parallel/learner.py:179: DataConversionWarning: A column-vector y was passed when a 1d array
  self.estimator.fit(xtemp, ytemp)
```

```
### TRAIN YOUR METALEARNER
```

```
log_reg = LogisticRegression(fit_intercept = False).fit(base_predictions, y_train)
sly_pred = log_reg.predict(super_learner.predict(X_test))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d arr
  y = column_or_1d(y, warn=True)
```

```
### INTERPRET COEFFICIENTS
```

```
log_reg.coef_
```

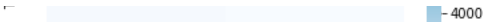
```
array([[-3.01112617,  9.82157052, -4.71526925, -4.27246556,  9.82157052]])
```

The coefficients in the output represent the weights assigned by the superlearner to each of the base estimators that were added to it. The results of the coefficient show that the logistic regression, KNeighborsClassifier, and AdaBoostClassifier will predict opposite. And Random forecast classifier and Decision tree classifier have greater impacts on the prediction accuracy of the superlearner because these two have larger absolute value of the coefficient, which are around 9.8.

```
### MAKE, SAVE AND VISUALIZE YOUR CONFUSION MATRIX
sly_pred = log_reg.predict(super_learner.predict(X_test))
y_true = y_test
cm_super = confusion_matrix(y_true, sly_pred)
```

```
class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_super, 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.) Create a bar chart comparing decision tree, bagged, boosted and super learner Sensitivities (Out of Sample)




```
# Compute the sensitivities and specificities
dt_sensitivity = cm_raw[1,1] / (cm_raw[1,1] + cm_raw[1,0])
dt_specificity = cm_raw[0,0] / (cm_raw[0,0] + cm_raw[0,1])

bag_sensitivity = cm_bag[1,1] / (cm_bag[1,1] + cm_bag[1,0])
bag_specificity = cm_bag[0,0] / (cm_bag[0,0] + cm_bag[0,1])

boost_sensitivity = cm_boost[1,1] / (cm_boost[1,1] + cm_boost[1,0])
boost_specificity = cm_boost[0,0] / (cm_boost[0,0] + cm_boost[0,1])

sl_sensitivity = cm_super[1,1] / (cm_super[1,1] + cm_super[1,0])
sl_specificity = cm_super[0,0] / (cm_super[0,0] + cm_super[0,1])



# Create bar charts
fig, ax = plt.subplots(1, 2, figsize=(10,5))

# Sensitivities chart
sensitivities = [dt_sensitivity, bag_sensitivity, boost_sensitivity, sl_sensitivity]
ax[0].bar(['Decision Tree', 'Bagged', 'Boosted', 'Super Learner'], sensitivities, color='b')
ax[0].set_title('Classifier Sensitivities')
ax[0].set_ylabel('Sensitivity')
ax[0].set_ylim([0,1])

# specificities chart
specificities = [dt_specificity, bag_specificity, boost_specificity, sl_specificity]
ax[1].bar(['Decision Tree', 'Bagged', 'Boosted', 'Super Learner'], specificities, color='orange')
ax[1].set_title('Classifier Specificities')
ax[1].set_ylabel('Specificity')
ax[1].set_ylim([0,1])
plt.show()
```