```
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.ensemble import AdaBoostClassifier

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
from mlens.ensemble import SuperLearner
from sklearn.preprocessing import StandardScaler
from sklearn.tree import plot_tree
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

```
[MLENS] backend: threading
```

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

adaboost.fit(X_scaled, y_train)

y_pred = adaboost.predict(X_test)
drive.mount('/content/gdrive/', force_remount = True)
```

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

```
df = pd.read_csv("/content/gdrive/MyDrive/stroke/bank.csv", sep = ";")
```

```
df
```

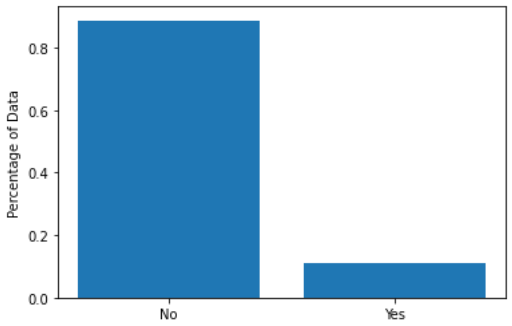
	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previo
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	

41188 rows × 21 columns

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

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

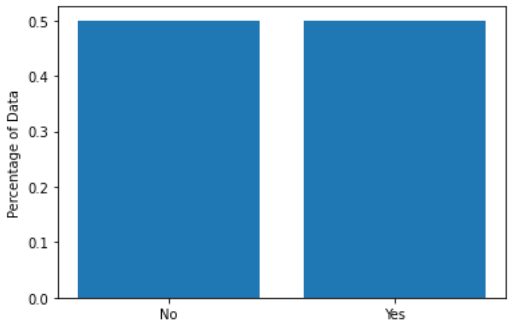
```
obs = len(y)
plt.bar(["No","Yes"],[len(y[y.yes==0])/obs,len(y[y.yes==1])/obs])
plt.ylabel("Percentage of Data")
plt.show()
```




```
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)
```

```
oversample = SMOTE()

X_scaled, y_train = oversample.fit_resample(X_scaled,y_train)
```

```
obs2 = len(y_train)
plt.bar(["No","Yes"],[len(y_train[y_train.yes==0])/obs2,len(y_train[y_train.yes==1])/obs2])
plt.ylabel("Percentage of Data")
plt.show()
```

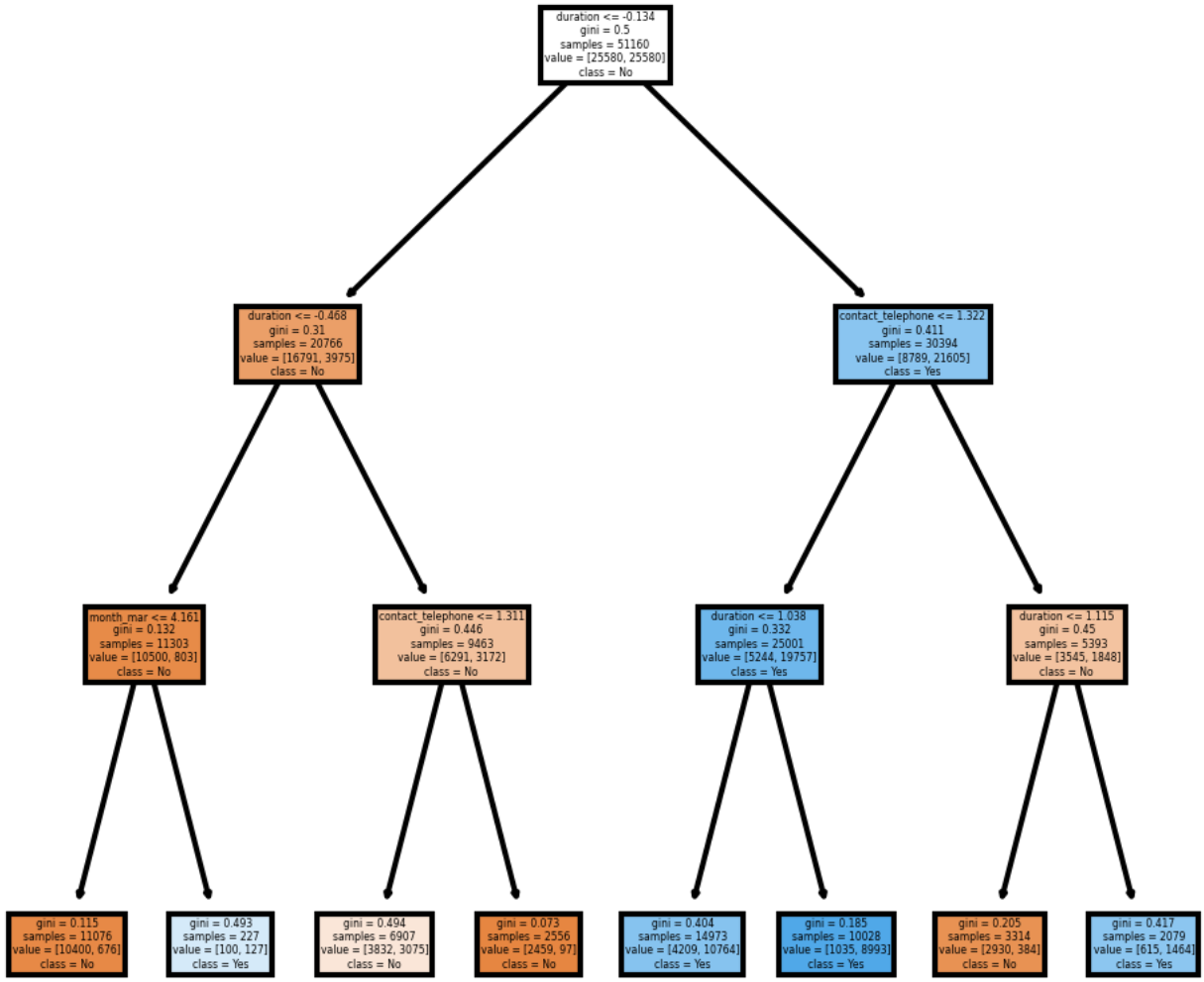



```
dtree = DecisionTreeClassifier(max_depth = 3)
dtree.fit(X_scaled, y_train)
```

```
▾ 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"])
```

```
[Text(0.5, 0.875, 'duration <= -0.134\ngini = 0.5\nsamples = 51160\nvalue = [25580, 25580]\nclass = No'),
 Text(0.25, 0.625, 'duration <= -0.468\ngini = 0.31\nsamples = 20766\nvalue = [16791, 3975]\nclass = No'),
 Text(0.125, 0.375, 'month_mar <= 4.161\ngini = 0.132\nsamples = 11303\nvalue = [10500, 803]\nclass = No'),
 Text(0.0625, 0.125, 'gini = 0.115\nsamples = 11076\nvalue = [10400, 676]\nclass = No'),
 Text(0.1875, 0.125, 'gini = 0.493\nsamples = 227\nvalue = [100, 127]\nclass = Yes'),
 Text(0.375, 0.375, 'contact_telephone <= 1.311\ngini = 0.446\nsamples = 9463\nvalue = [6291, 3172]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.494\nsamples = 6907\nvalue = [3832, 3075]\nclass = No'),
 Text(0.4375, 0.125, 'gini = 0.073\nsamples = 2556\nvalue = [2459, 97]\nclass = No'),
 Text(0.75, 0.625, 'contact_telephone <= 1.322\ngini = 0.411\nsamples = 30394\nvalue = [8789, 21605]\nclass = Yes'),
 Text(0.625, 0.375, 'duration <= 1.038\ngini = 0.332\nsamples = 25001\nvalue = [5244, 19757]\nclass = Yes'),
 Text(0.5625, 0.125, 'gini = 0.404\nsamples = 14973\nvalue = [4209, 10764]\nclass = Yes'),
 Text(0.6875, 0.125, 'gini = 0.185\nsamples = 10028\nvalue = [1035, 8993]\nclass = Yes'),
 Text(0.875, 0.375, 'duration <= 1.115\ngini = 0.45\nsamples = 5393\nvalue = [3545, 1848]\nclass = No'),
 Text(0.8125, 0.125, 'gini = 0.205\nsamples = 3314\nvalue = [2930, 384]\nclass = No'),
 Text(0.9375, 0.125, 'gini = 0.417\nsamples = 2079\nvalue = [615, 1464]\nclass = Yes')]
```



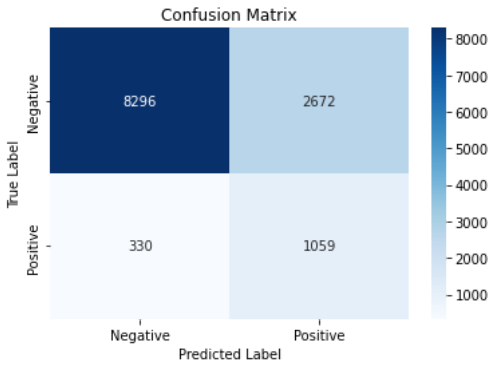

```
y_pred = dtree.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)

accuracy = accuracy_score(y_true, y_pred)
print("DT Accuracy:", accuracy)
```

```
DT Accuracy: 0.7638585417172453
```

```
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)

y_pred = bagging.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Bagged DT Accuracy:", accuracy)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_bagging.py:802: DataConversionWarning: A column-vector y was passed
  y = column_or_1d(y, warn=True)
Bagged DT Accuracy: 0.7570607752690782
```

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



class_labels = ['Negative', 'Positive']

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()
```




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

adaboost.fit(X_scaled, y_train)

y_pred = adaboost.predict(X_test)
```

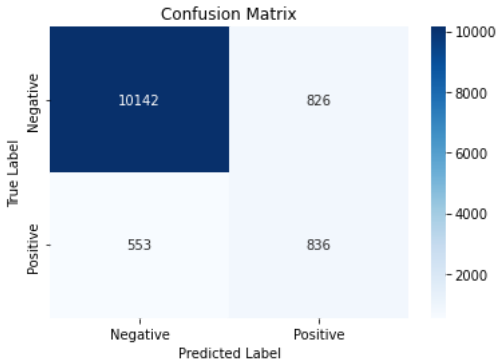
```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimat
  warnings.warn(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

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

```
accuracy = accuracy_score(y_true, y_pred)
print("Bossted DT Accuracy:", accuracy)
```

```
Bossted DT Accuracy: 0.8884033341425912
```

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

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()
```




```
base_estimators = [
    LogisticRegression(),
    RandomForestClassifier(),
    QDA(),
    KNeighborsClassifier(n_neighbors=5),
    SVC()
]

super_learner = SuperLearner(folds=10, random_state =42)

super_learner.add(base_estimators)
```

```
super_learner.fit(X_scaled, y_train)
```

```
base_predictions = super_learner.predict(X_scaled)

base_predictions
```

```
  self.estimator.fit(xtemp, ytemp)
/usr/local/lib/python3.8/dist-packages/mlens/parallel/learner.py:179: DataConversionWarning: A column-vector y was passed
  self.estimator.fit(xtemp, ytemp)
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was pas
  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 pas
  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 pas
  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 pas
  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 pas
  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 pas
  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 pas
  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 pas
  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 pas
  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 pas
  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 pas
  y = column_or_1d(y, warn=True)
array([[0., 1., 1., 0., 1.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       ...,
       [1., 1., 1., 1., 1.],
```

```
log_reg = LogisticRegression(fit_intercept = False).fit(base_predictions,y_train)
y_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
  y = column_or_1d(y, warn=True)
```

```
log_reg.coef_
```

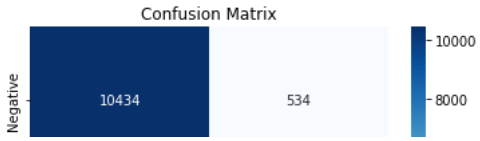
```
array([[ 0.12430991, -0.47044306, -7.47490695, 13.59458739,  1.90069226]])
```

The coefficients represent the magnitude that each model prediction has on the final prediction and the sign indicates the direction of the prediction.

KNN5 has the strongest positive effect on the final prediction while QDA has the strongest negative effect on the final prediction. Random Forest has a weak negative effect on the final prediction. The logit model has a weak positive effect on the final prediction and SVC has the second strongest positive effect on the final prediction.

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

accuracy = accuracy_score(y_true, y_pred)
print("SL Accuracy:", accuracy)

class_labels = ['Negative', 'Positive']


sns.heatmap(cm_SL, 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()
```

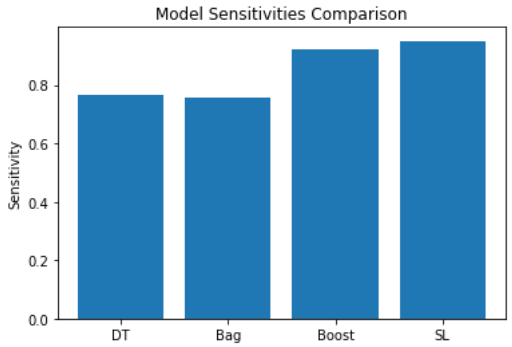
```
SL Accuracy: 0.8915594399935259
```



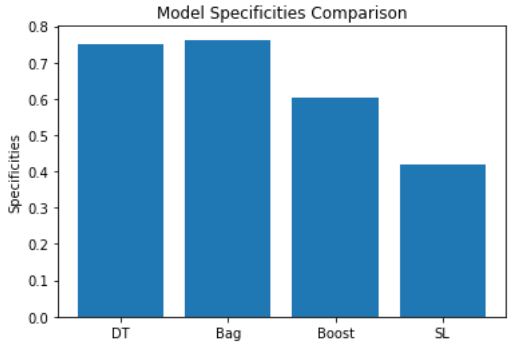

```
sensitivity1 = cm_SL[0,0]/(cm_SL[0,0]+cm_SL[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm_SL[1,1]/(cm_SL[1,0]+cm_SL[1,1])
print('Specificity : ', specificity1)
```

```
Sensitivity :  0.9513129102844639
Specificity :  0.4197264218862491
```

```
sen_raw = cm_raw[0,0]/(cm_raw[0,0]+cm_raw[0,1])
spec_raw = cm_raw[1,1]/(cm_raw[1,0]+cm_raw[1,1])


sen_bag = cm_bag[0,0]/(cm_bag[0,0]+cm_bag[0,1])
spec_bag = cm_bag[1,1]/(cm_bag[1,0]+cm_bag[1,1])


sen_boost = cm_boost[0,0]/(cm_boost[0,0]+cm_boost[0,1])
spec_boost = cm_boost[1,1]/(cm_boost[1,0]+cm_boost[1,1])


sen_SL = cm_SL[0,0]/(cm_SL[0,0]+cm_SL[0,1])
spec_SL = cm_SL[1,1]/(cm_SL[1,0]+cm_SL[1,1])


plt.title("Model Sensitivities Comparison")
plt.bar(["DT","Bag","Boost","SL"],[sen_raw,sen_bag,sen_boost,sen_SL])
plt.ylabel("Sensitivity")
plt.show()
```




```
plt.title("Model Specificities Comparison")
plt.bar(["DT","Bag","Boost","SL"],[spec_raw,spec_bag,spec_boost,spec_SL])
plt.ylabel("Specificities")
plt.show()
```



✓ 0s completed at 4:16 PM