Assignment1_pt3

February 13, 2023

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

1 Read and check data

[ ]:
```
df = pd.read_csv('data.csv')
df.head(5)
```

[ ]:

	Purchase	SUS	Duration	Gender	ASR_Error	Intent_Error
0	1	84	254	0	3	2
1	0	58	247	0	6	9
2	0	56	125	1	6	8
3	0	55	22	0	11	7
4	1	95	262	0	2	3

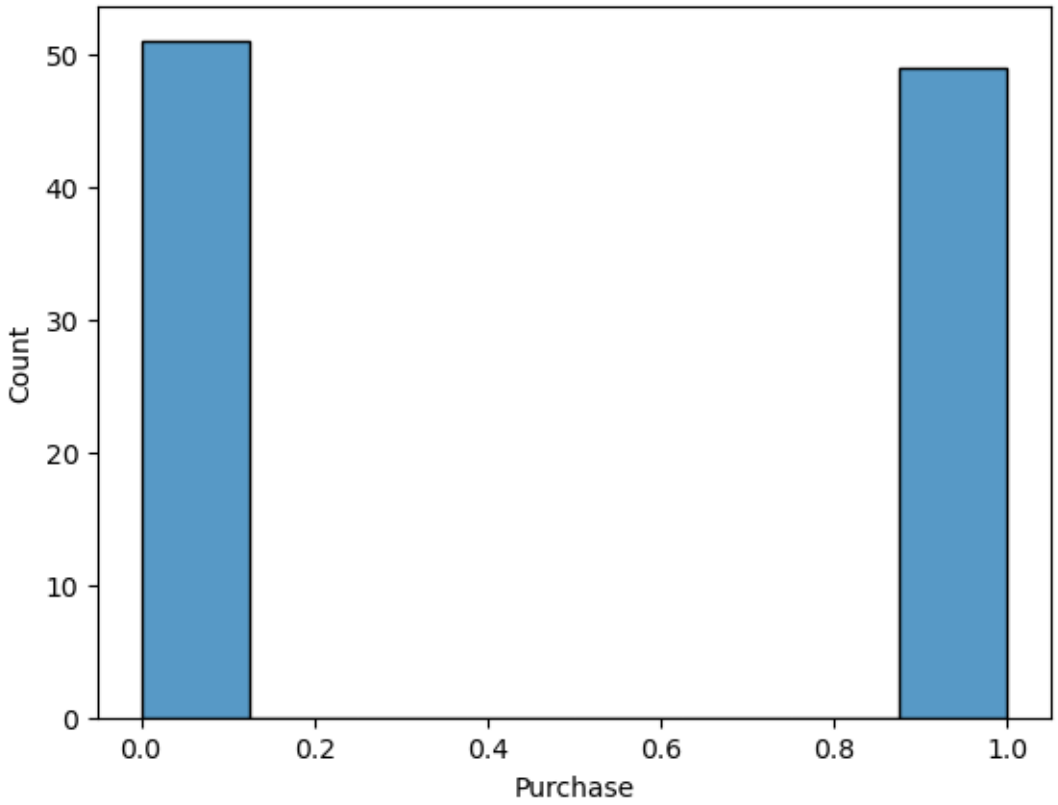
[ ]:
```
df.isna().sum()
```

[ ]:
```
Purchase        0
SUS             0
Duration        0
Gender          0
ASR_Error       0
Intent_Error    0
dtype: int64
```

2 See distribution of Purchases

[ ]:
```
# histogram of System Usability Survery (SUS) score
sns.histplot(data=df['Purchase'], palette='bright')
```

[ ]:
```
<AxesSubplot: xlabel='Purchase', ylabel='Count'>
```



3 Prep and label data for model training

[ ]:
```
# label
y = df['Purchase'].to_numpy()
y
```

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

[ ]:
```
# data
X = df.drop(['Purchase', 'SUS'], axis = 1).to_numpy()
X
```

[ ]:
```
array([[254,   0,   3,   2],
       [247,   0,   6,   9],
       [125,   1,   6,   8],
       [ 22,   0,  11,   7],
```

```
[262,   0,   2,   3],
[113,   1,   8,   9],
[ 91,   1,   6,   3],
[ 46,   0,   6,  11],
[132,   0,   9,   9],
[190,   0,  11,   4],
[146,   0,   9,   7],
[226,   0,   4,   4],
[ 54,   1,   7,   9],
[104,   0,   2,   3],
[292,   0,   7,   3],
[126,   0,   8,   9],
[152,   1,   3,   5],
[221,   0,   3,   4],
[ 51,   0,   9,   9],
[230,   0,   6,   8],
[ 72,   1,   6,   8],
[284,   0,   6,  11],
[168,   0,   2,   4],
[194,   0,   1,   3],
[130,   1,  10,   1],
[227,   0,   9,   3],
[161,   1,   7,   8],
[262,   1,   2,   3],
[257,   1,   9,  11],
[ 57,   1,   8,   3],
[138,   0,   8,   9],
[ 33,   1,   9,   4],
[260,   0,   1,   2],
[178,   1,   3,   4],
[153,   0,   2,   4],
[151,   0,   2,   3],
[ 67,   0,   8,   9],
[284,   1,   2,   2],
[ 96,   1,   3,   1],
[ 31,   0,   1,   3],
[100,   0,  12,   9],
[200,   0,   2,   3],
[ 24,   1,   1,   2],
[169,   0,   1,   3],
[110,   0,   1,   3],
[184,   1,   2,   3],
[234,   1,   2,   5],
[ 90,   1,   5,   3],
[202,   1,   3,   3],
[254,   0,   3,   3],
[259,   0,   9,   6],
```

```
[193,   1,  11,   5],
[258,   0,   6,   5],
[117,   1,   6,   7],
[297,   1,   3,   4],
[218,   1,  12,   9],
[119,   0,   0,   8],
[ 54,   1,   8,   6],
[143,   1,   0,   4],
[300,   1,   9,  11],
[ 43,   0,   3,   4],
[ 28,   0,   7,   6],
[357,   1,   9,   4],
[ 21,   0,  13,   7],
[100,   0,   7,   9],
[303,   1,   9,   8],
[ 56,   1,   6,  10],
[205,   0,   5,  12],
[104,   0,   4,   1],
[342,   0,   1,   3],
[ 78,   0,   3,   2],
[ 56,   0,   6,   5],
[297,   1,   2,   1],
[ 56,   0,   5,   4],
[365,   0,   4,   3],
[275,   0,  11,   6],
[198,   1,   9,   9],
[224,   0,  12,  12],
[326,   0,   7,   7],
[ 23,   0,  14,   9],
[ 78,   1,   9,  10],
[108,   0,   6,  11],
[326,   0,  13,   7],
[ 61,   0,  10,   8],
[ 48,   1,   1,   5],
[ 90,   0,  11,   4],
[ 87,   1,   0,   1],
[ 65,   1,   2,   2],
[ 22,   1,   3,   0],
[ 10,   1,   9,   9],
[ 45,   0,   5,   8],
[ 69,   1,   4,   0],
[ 57,   0,   1,   1],
[208,   1,   0,   4],
[197,   0,  11,  10],
[358,   0,  13,   7],
[ 71,   0,   3,   0],
[ 34,   1,   0,   9],
```

```
       [ 49,   1,   4,   1],
       [213,   0,   1,   4]])
```

4 Data Scaling

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn import linear_model
from sklearn.model_selection import train_test_split

scale = StandardScaler()
scaled_X = scale.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(scaled_X, y, test_size = 0.
 ↪3)
y_test
```

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

5 Import Classifiers and Metrics from Sklearn

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix,␣
 ↪roc_curve, auc, classification_report

import matplotlib.pyplot as plt
```

[ ]:
```
lc = LogisticRegression()
svc = SVC(probability=True)
nbc = GaussianNB()
rfc = RandomForestClassifier()
```

6 Train ML model with training dataset with model.fit() function

[ ]:
```
lc.fit(X_train, y_train)
svc.fit(X_train, y_train)
nbc.fit(X_train, y_train)
rfc.fit(X_train, y_train)
```

[ ]:
```
RandomForestClassifier()
```

7 Test model with test dataset

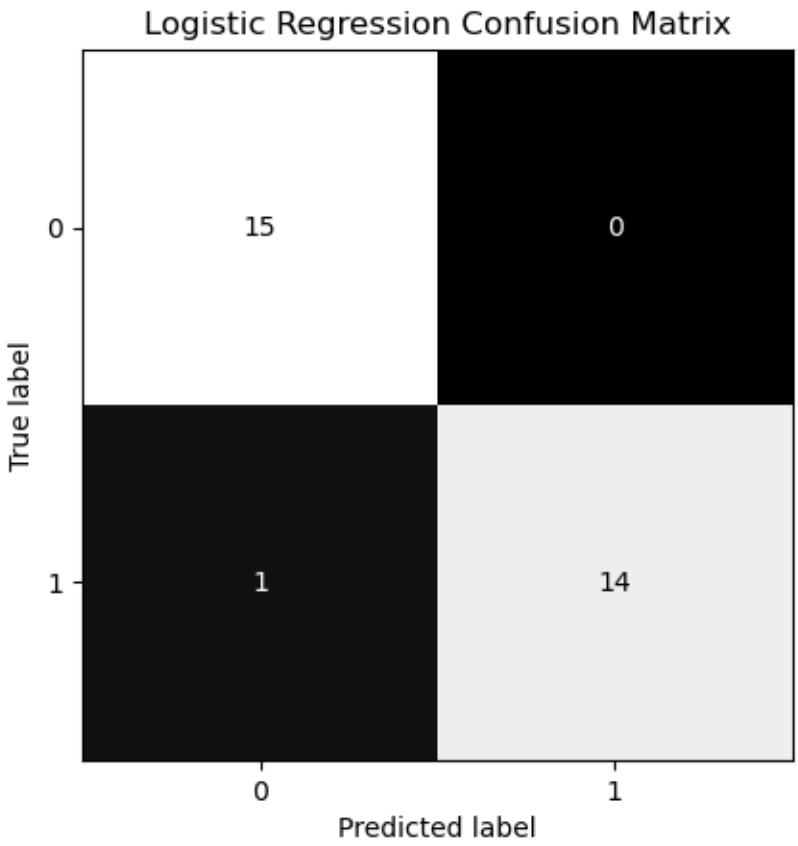
[ ]:
```
y_lc_predicted = lc.predict(X_test)
y_lc_pred_proba = lc.predict_proba(X_test)
# print(y_lc_pred_proba)

y_svc_predicted = svc.predict(X_test)
y_svc_pred_proba = svc.predict_proba(X_test)

y_nbc_predicted = nbc.predict(X_test)
y_nbc_pred_proba = nbc.predict_proba(X_test)

y_rfc_predicted = rfc.predict(X_test)
y_rfc_pred_proba = rfc.predict_proba(X_test)
```

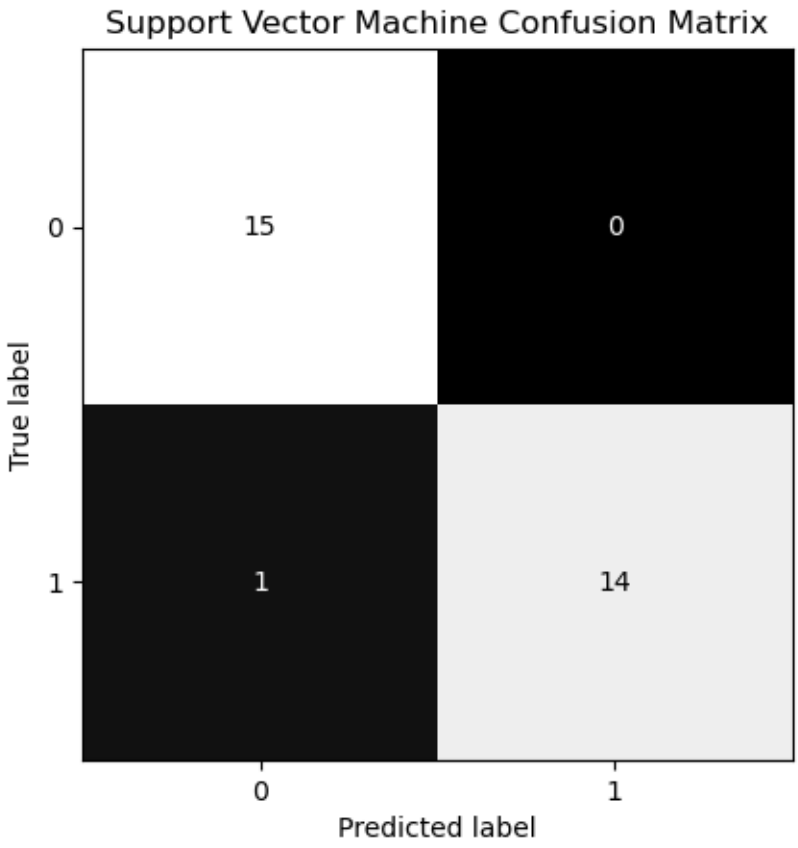
[ ]:
```
### model evaluation with confusion matrix and ROC curve ###

# print classification reports for each model
print("Logistic Regression")
print(classification_report(y_test, y_lc_predicted))
print("Support Vector Classifier")
print(classification_report(y_test, y_svc_predicted))
print("Gaussian Naive Bayes")
print(classification_report(y_test, y_nbc_predicted))
print("Random Forest Classifier")
print(classification_report(y_test, y_rfc_predicted))

# list models, predictions, and probabilities
models = ['Logistic Regression', 'Support Vector Machine', 'Naive Bayes␣
 ↪Classifier', 'Random Forest Classifier']
predictions = [y_lc_predicted, y_svc_predicted, y_nbc_predicted,␣
 ↪y_rfc_predicted]
pred_probabilities = [y_lc_pred_proba, y_svc_pred_proba, y_nbc_pred_proba,␣
 ↪y_rfc_pred_proba]

# display confusion matrix for each model
plot = 1
for model, prediction, pred_proba in zip(models, predictions,␣
 ↪pred_probabilities):
    disp = ConfusionMatrixDisplay(confusion_matrix(y_test.ravel(), prediction))
    disp.plot(
        include_values=True,
        cmap='gray',
        colorbar=False
    )
    disp.ax_.set_title(f"{model} Confusion Matrix")
```

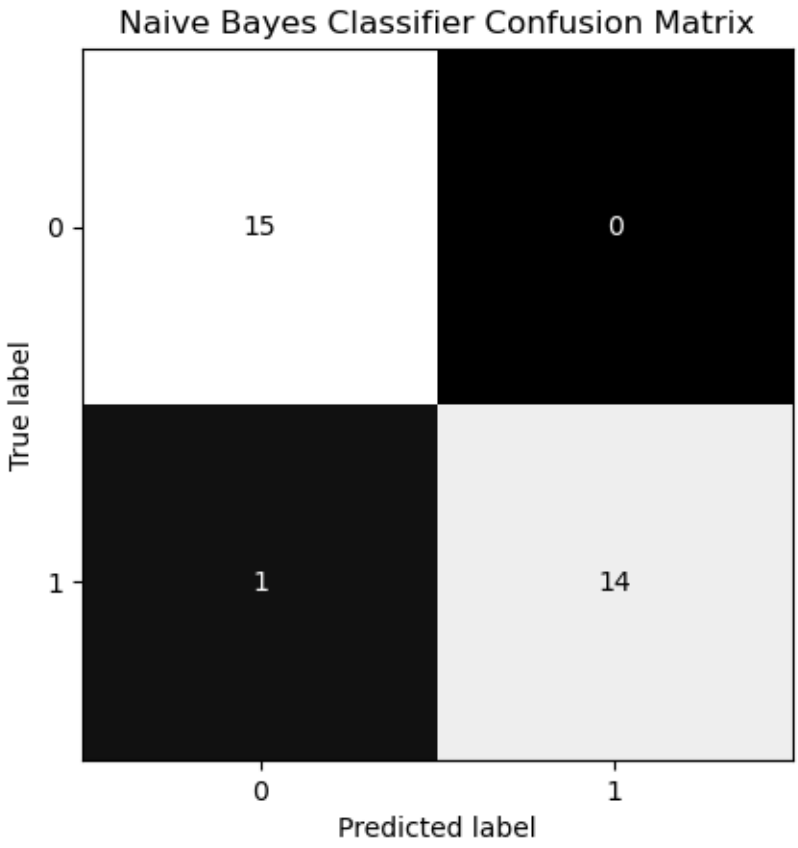
```
# display ROC Curve of each model
plt.figure(figsize=(30, 15))
plt.suptitle("ROC Curves")
plot_index = 1

for model, prediction, pred_proba in zip(models, predictions,␣
 ↪pred_probabilities):
    # fpr - false pos rate, tpr - true pos rate, auc - area under curve (ROC)
    fpr, tpr, thresholds = roc_curve(y_test, pred_proba[:, 1])
    auc_score = auc(fpr, tpr)
    plt.subplot(3, 2, plot_index)
    plt.plot(fpr, tpr, 'r', label='ROC curve')
    plt.title(f'Roc Curve - {model} - [AUC - {auc_score}]', fontsize=14)
    plt.xlabel('FPR', fontsize=12)
    plt.ylabel('TPR', fontsize=12)
    plt.legend()
    plot_index += 1
plt.show()
```

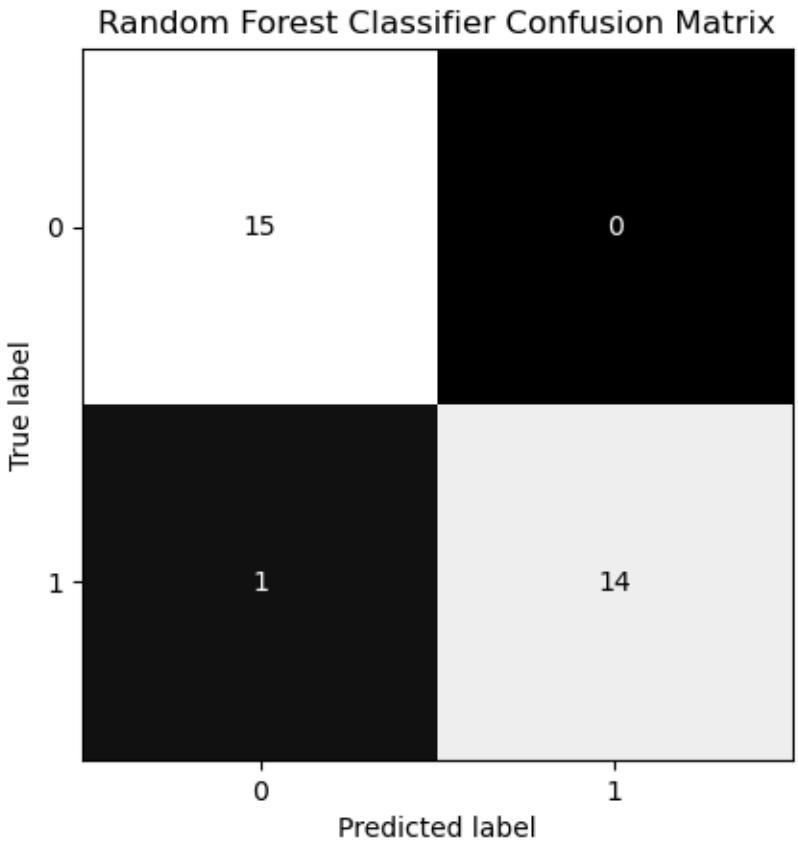
```
Logistic Regression
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.93      0.97        15

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30

Support Vector Classifier
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.93      0.97        15

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30

Gaussian Naive Bayes
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.93      0.97        15

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
```

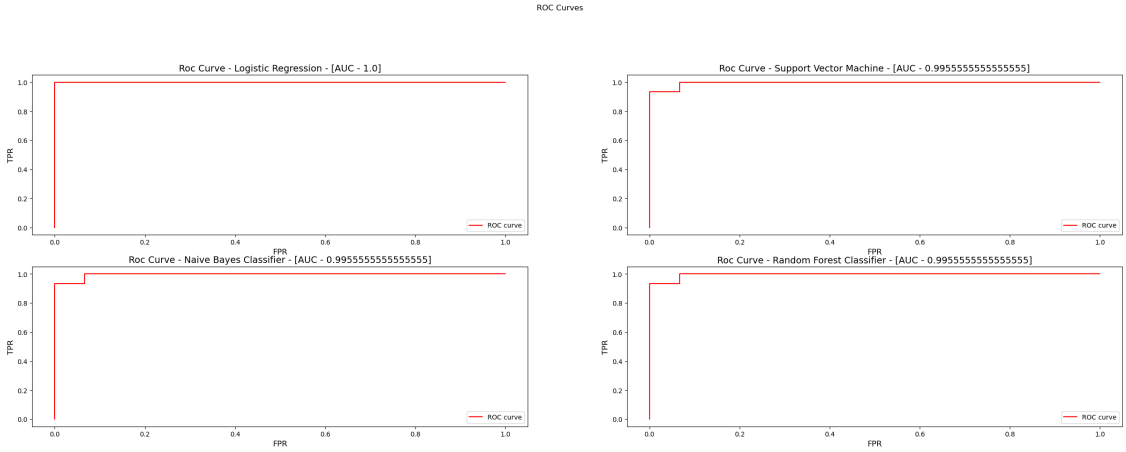
```
weighted avg       0.97      0.97      0.97        30

Random Forest Classifier
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.93      0.97        15

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```




Support Vector Machine Confusion Matrix
15
0
1
14
0
1
True label
Predicted label


Naive Bayes Classifier Confusion Matrix
15
0
1
14
0
1
True label
Predicted label





[ ]:
```
# use SMOTE for imbalanced classification

from imblearn.over_sampling import SMOTE
```

```
oversample = SMOTE()
over_sampled_X_train, over_sampled_y_train = oversample.fit_resample(X_train,␣
↪y_train)
```

[ ]:
```
lc = LogisticRegression()
svc = SVC(probability=True)
nbc = GaussianNB()
rfc = RandomForestClassifier()
```

[ ]:
```
lc.fit(over_sampled_X_train, over_sampled_y_train)
svc.fit(over_sampled_X_train, over_sampled_y_train)
nbc.fit(over_sampled_X_train, over_sampled_y_train)
rfc.fit(over_sampled_X_train, over_sampled_y_train)
```

[ ]:
```
RandomForestClassifier()
```

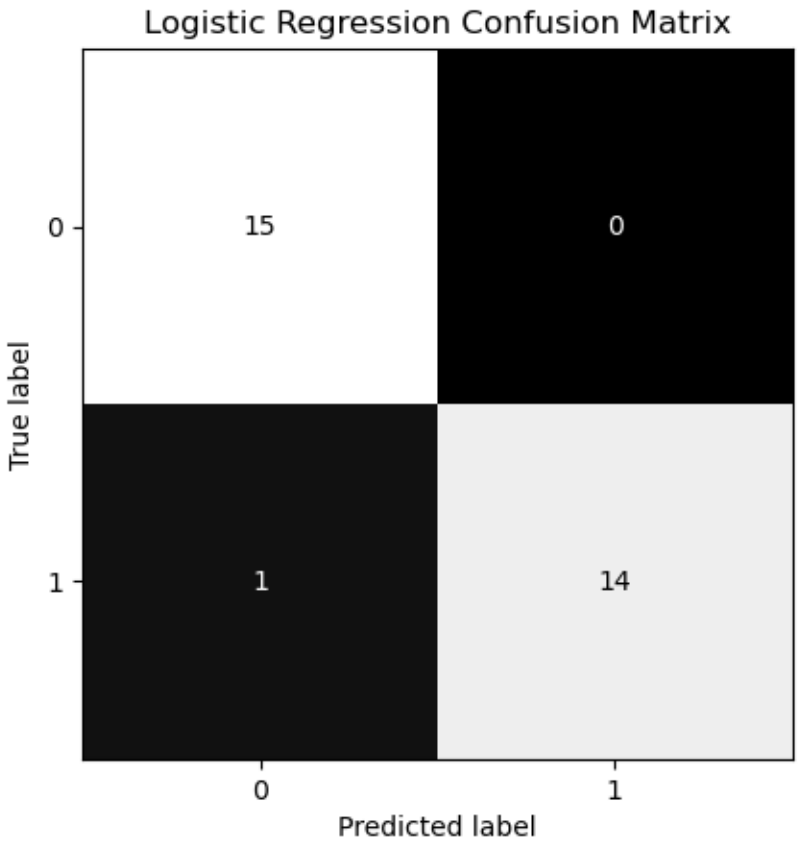
[ ]:
```
y_lc_predicted = lc.predict(X_test)
y_lc_pred_proba = lc.predict_proba(X_test)

y_svc_predicted = svc.predict(X_test)
y_svc_pred_proba = svc.predict_proba(X_test)

y_nbc_predicted = nbc.predict(X_test)
y_nbc_pred_proba = nbc.predict_proba(X_test)

y_rfc_predicted = rfc.predict(X_test)
y_rfc_pred_proba = rfc.predict_proba(X_test)

print("Logistic Regression")
print(classification_report(y_test, y_lc_predicted))
print("Support Vector Classifier")
print(classification_report(y_test, y_svc_predicted))
print("Gaussian Naive Bayes")
print(classification_report(y_test, y_nbc_predicted))
print("Random Forest Classifier")
print(classification_report(y_test, y_rfc_predicted))
```

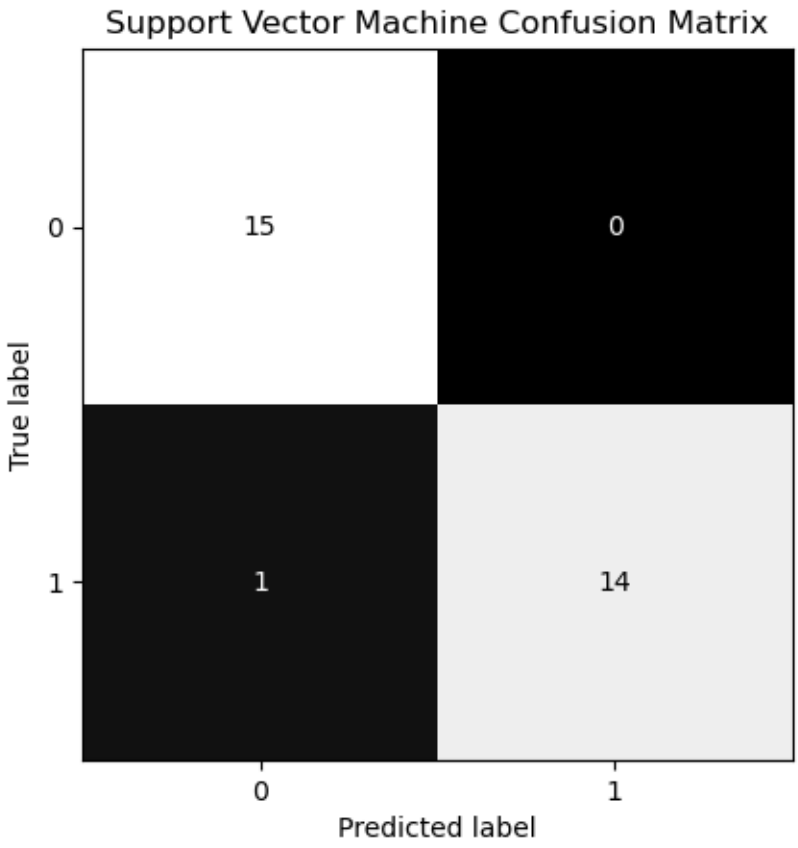
```
Logistic Regression
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.93      0.97        15

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

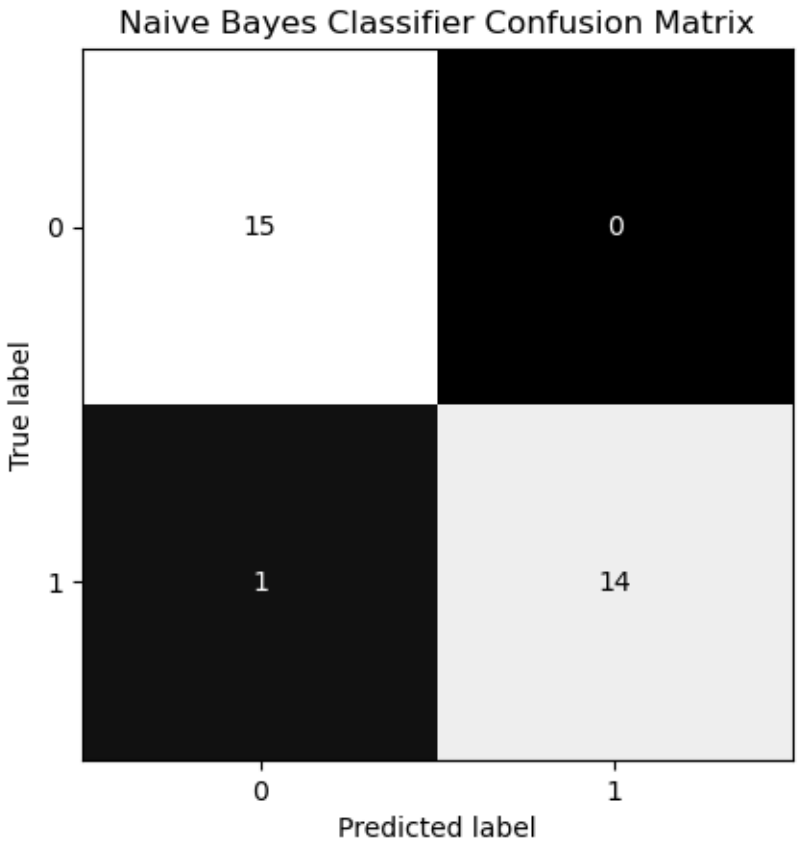
```
Support Vector Classifier
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.93      0.97        15

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30

Gaussian Naive Bayes
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.93      0.97        15

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30

Random Forest Classifier
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        15
           1       0.93      0.93      0.93        15

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30
```

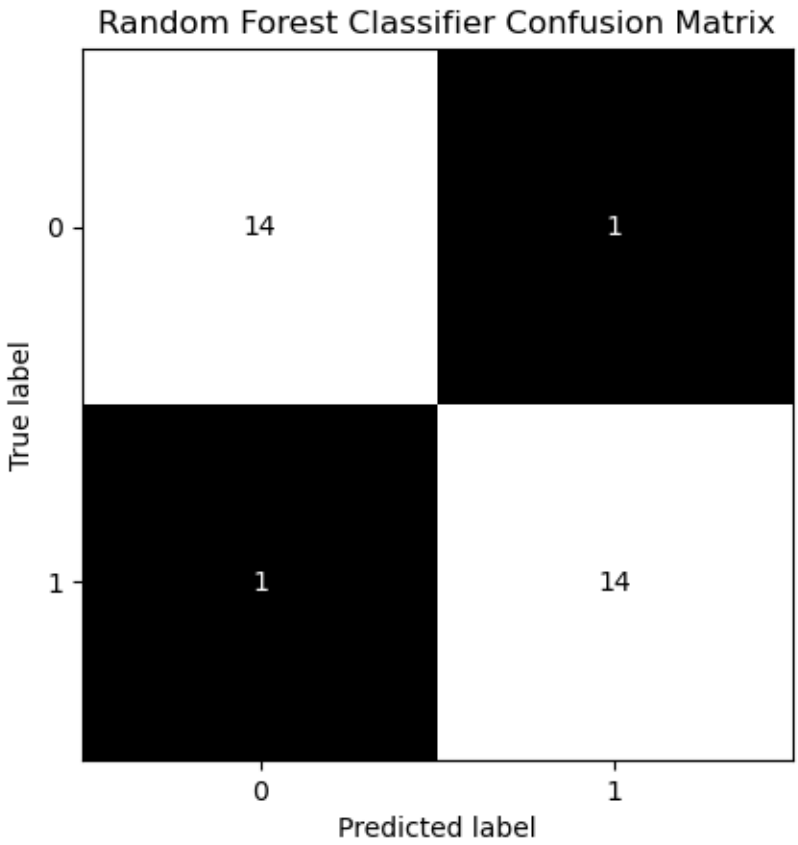
[ ]:
```
models = ['Logistic Regression', 'Support Vector Machine', 'Naive Bayes␣
 ↪Classifier', 'Random Forest Classifier']
predictions = [y_lc_predicted, y_svc_predicted, y_nbc_predicted,␣
 ↪y_rfc_predicted]
pred_probabilities = [y_lc_pred_proba, y_svc_pred_proba, y_nbc_pred_proba,␣
 ↪y_rfc_pred_proba]

plot = 1

for model, prediction, pred_proba in zip(models, predictions,␣
 ↪pred_probabilities):
    disp = ConfusionMatrixDisplay(confusion_matrix(y_test.ravel(), prediction))
    disp.plot(
        include_values=True,
        cmap='gray',
        colorbar=False
```

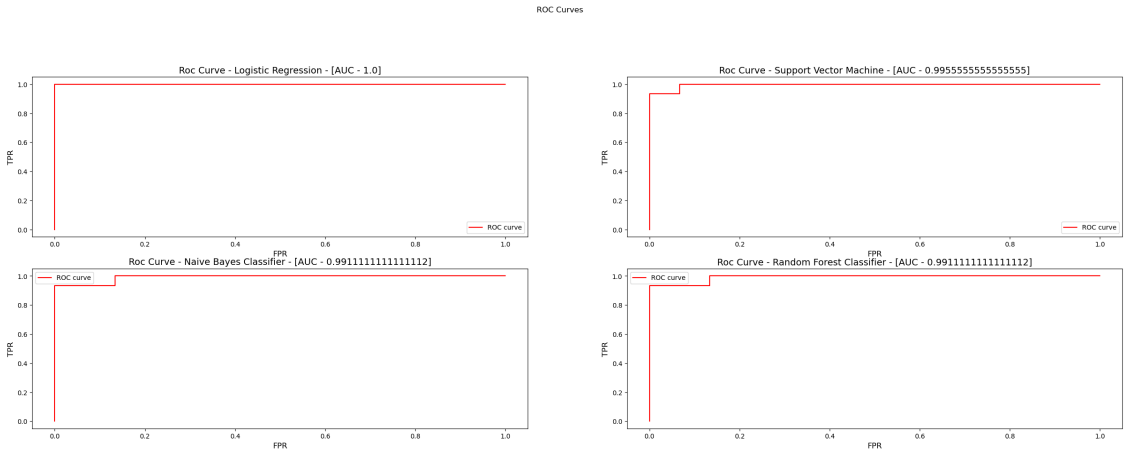
```
    )
    disp.ax_.set_title(f"{model} Confusion Matrix")

plt.figure(figsize=(30, 15))
plt.suptitle("ROC Curves")
plot_index = 1

for model, prediction, pred_proba in zip(models, predictions,␣
 ↪pred_probabilities):
    fpr, tpr, thresholds = roc_curve(y_test, pred_proba[:, 1])
    auc_score = auc(fpr, tpr)
    plt.subplot(3, 2, plot_index)
    plt.plot(fpr, tpr, 'r', label='ROC curve')
    # pyplot.figure(figsize=(5, 5))
    plt.title(f'Roc Curve - {model} - [AUC - {auc_score}]', fontsize=14)
    plt.xlabel('FPR', fontsize=12)
    plt.ylabel('TPR', fontsize=12)
    plt.legend()
    plot_index += 1
plt.show()
```




Support Vector Machine Confusion Matrix
15
0
1
14
0
1
True label
Predicted label


Naive Bayes Classifier Confusion Matrix
15
0
1
14
0
1
True label
Predicted label


Random Forest Classifier Confusion Matrix
True label
Predicted label
0
1
14
1
1
14
0
1


ROC Curves
Roc Curve - Logistic Regression - [AUC - 1.0]
Roc Curve - Support Vector Machine - [AUC - 0.9955555555555555]
Roc Curve - Naive Bayes Classifier - [AUC - 0.9911111111111112]
Roc Curve - Random Forest Classifier - [AUC - 0.9911111111111112]
TPR
FPR
ROC curve