GERON - END-TO-END PROJECT

Kwabena Asabere

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

```
df = pd.read_csv(r"C:\Users\KAsab\Downloads\alzheimers_disease_data.csv")
```

```
df.head()
```

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	Ph
0	4751	73	0	0	2	22.927749	0	13.297218	6.3
1	4752	89	0	0	0	26.827681	0	4.542524	7.0
2	4753	73	0	3	1	17.795882	0	19.555085	7.8
3	4754	74	1	0	1	33.800817	1	12.209266	8.4
4	4755	89	0	0	0	20.716974	0	18.454356	6.3

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2149 entries, 0 to 2148
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   PatientID                  2149 non-null   int64
 1   Age                        2149 non-null   int64
 2   Gender                     2149 non-null   int64
 3   Ethnicity                  2149 non-null   int64
 4   EducationLevel             2149 non-null   int64
 5   BMI                        2149 non-null   float64
```

```
 6   Smoking                    2149 non-null   int64
 7   AlcoholConsumption         2149 non-null   float64
 8   PhysicalActivity           2149 non-null   float64
 9   DietQuality                2149 non-null   float64
 10  SleepQuality               2149 non-null   float64
 11  FamilyHistoryAlzheimers    2149 non-null   int64
 12  CardiovascularDisease      2149 non-null   int64
 13  Diabetes                   2149 non-null   int64
 14  Depression                 2149 non-null   int64
 15  HeadInjury                 2149 non-null   int64
 16  Hypertension               2149 non-null   int64
 17  SystolicBP                 2149 non-null   int64
 18  DiastolicBP                2149 non-null   int64
 19  CholesterolTotal           2149 non-null   float64
 20  CholesterolLDL             2149 non-null   float64
 21  CholesterolHDL             2149 non-null   float64
 22  CholesterolTriglycerides   2149 non-null   float64
 23  MMSE                       2149 non-null   float64
 24  FunctionalAssessment       2149 non-null   float64
 25  MemoryComplaints           2149 non-null   int64
 26  BehavioralProblems         2149 non-null   int64
 27  ADL                        2149 non-null   float64
 28  Confusion                  2149 non-null   int64
 29  Disorientation             2149 non-null   int64
 30  PersonalityChanges         2149 non-null   int64
 31  DifficultyCompletingTasks  2149 non-null   int64
 32  Forgetfulness              2149 non-null   int64
 33  Diagnosis                  2149 non-null   int64
 34  DoctorInCharge             2149 non-null   object
dtypes: float64(12), int64(22), object(1)
memory usage: 587.7+ KB
```

```
df.drop(columns =[ "PatientID","DoctorInCharge"],inplace = True)
```

```
df.head()
```

	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivi
0	73	0	0	2	22.927749	0	13.297218	6.327112
1	89	0	0	0	26.827681	0	4.542524	7.619885
2	73	0	3	1	17.795882	0	19.555085	7.844988
3	74	1	0	1	33.800817	1	12.209266	8.428001

	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivi
4	89	0	0	0	20.716974	0	18.454356	6.310461

```
heart = pd.read_csv(r"C:\Users\KAsab\Desktop\Analysis_Workshop\data\south_africa_heart.csv")
```

```
heart.head()
```

	CLASS	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age
0	1	160	12.00	5.73	23.11	1	49	25.30	97.20	52
1	1	144	0.01	4.41	28.61	0	55	28.87	2.06	63
2	-1	118	0.08	3.48	32.28	1	52	29.14	3.81	46
3	1	170	7.50	6.41	38.03	1	51	31.99	24.26	58
4	1	134	13.60	3.50	27.78	1	60	25.99	57.34	49

```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 462 entries, 0 to 461
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   CLASS      462 non-null    int64
 1   sbp        462 non-null    int64
 2   tobacco    462 non-null    float64
 3   ldl        462 non-null    float64
 4   adiposity  462 non-null    float64
 5   famhist    462 non-null    int64
 6   typea      462 non-null    int64
 7   obesity    462 non-null    float64
 8   alcohol    462 non-null    float64
 9   age        462 non-null    int64
dtypes: float64(5), int64(5)
memory usage: 36.2 KB
```

```
heart_target= heart["CLASS"]
heart_features = heart.iloc[:,1:9]
```

```
from sklearn.model_selection import train_test_split,cross_val_score,cross_val_predict
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix,precision_score,f1_score
from sklearn.preprocessing import StandardScaler,MinMaxScaler,OneHotEncoder
from sklearn.pipeline import Pipeline,make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier
```

```
heart_features_train,heart_features_test,heart_target_train,heart_target_test =
train_test_split(heart_features,heart_target, random_state = 42)
```

```
y_train,y_test = heart_target_train,heart_target_test
```

```
X_train,X_test = heart_features_train,heart_features_test
```

```
rfc= RandomForestClassifier(random_state = 42)
lrc = LogisticRegression(random_state = 42)
dmc = DummyClassifier(random_state = 42)
```

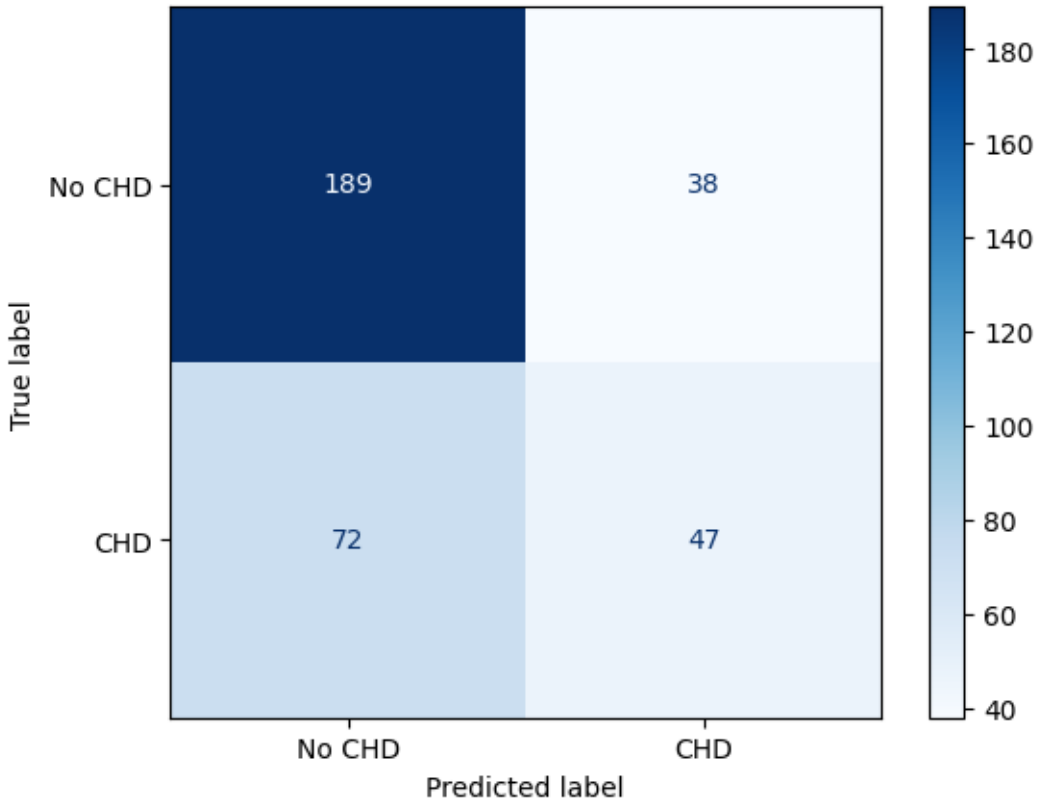
```
y_train_pred = cross_val_predict(rfc,X_train,y_train, cv = 3)
```

```
cm = confusion_matrix(y_train,y_train_pred)
```

```
print(f"Confusion Matrix:",cm)
```

```
Confusion Matrix: [[189  38]
 [ 72  47]]
```

```
disp = ConfusionMatrixDisplay(confusion_matrix = cm,display_labels = ["No CHD","CHD"])
disp.plot(cmap = plt.cm.Blues)
plt.show()
```




```
precision_score(y_train,y_train_pred)
```

```
0.5529411764705883
```

```
accuracy_score(y_train,y_train_pred)
```

```
0.6820809248554913
```

```
lr_model = make_pipeline(StandardScaler(),LogisticRegression())
```

```
lr_model.fit(X_train,y_train)
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
```

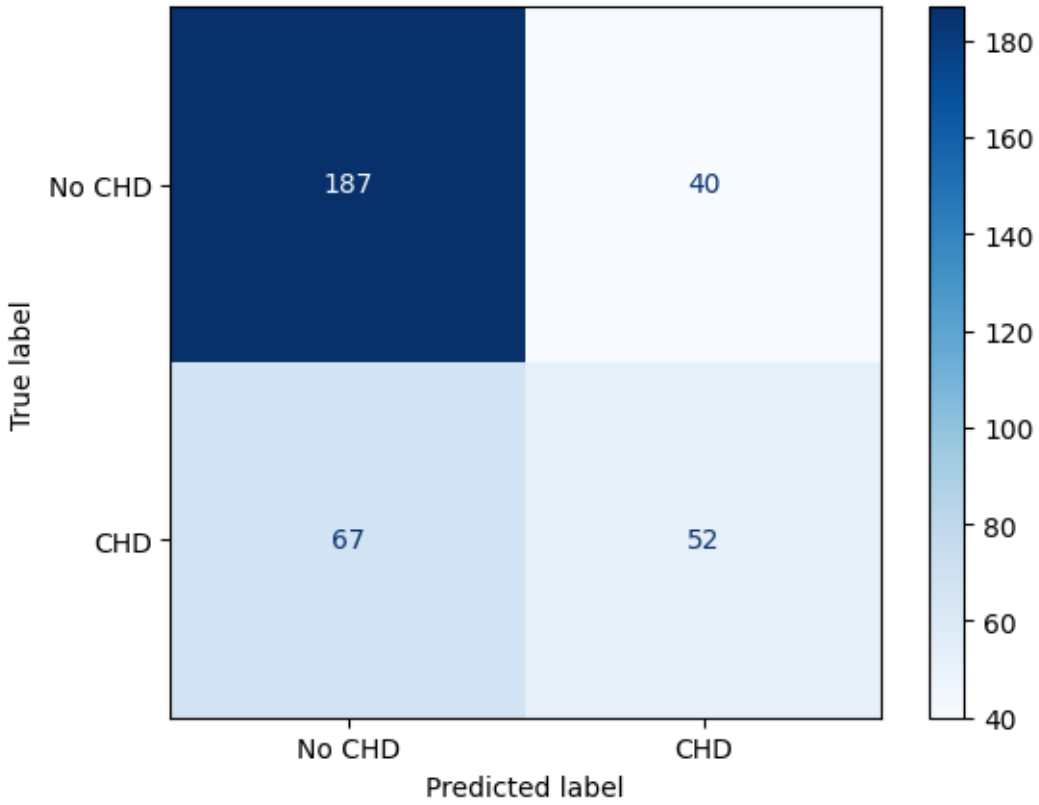
```
y_pred_lr = cross_val_predict(lr_model,X_train,y_train,cv = 3)
```

```
y_scores = cross_val_predict(lr_model,X_train,y_train,cv = 3,method = "decision_function") #
y_proba =  cross_val_predict(lr_model,X_train,y_train,cv = 3,method = "predict_proba")[:,1]#
```

```
pd.Series(y_scores).describe()
```

```
count    346.000000
mean      -0.794912
std        1.236170
min       -6.868398
25%       -1.641016
50%       -0.814414
75%        0.063084
max        3.691798
dtype: float64
```

```
cm_lr = confusion_matrix(y_train,y_pred_lr)
disp = ConfusionMatrixDisplay(confusion_matrix = cm_lr,display_labels = ["No CHD","CHD"])
disp.plot(cmap = plt.cm.Blues)
plt.show()
```




```
precision_score(y_train,y_pred_lr)
```

```
0.5652173913043478
```

```
recall_score(y_train,y_pred_lr)
```

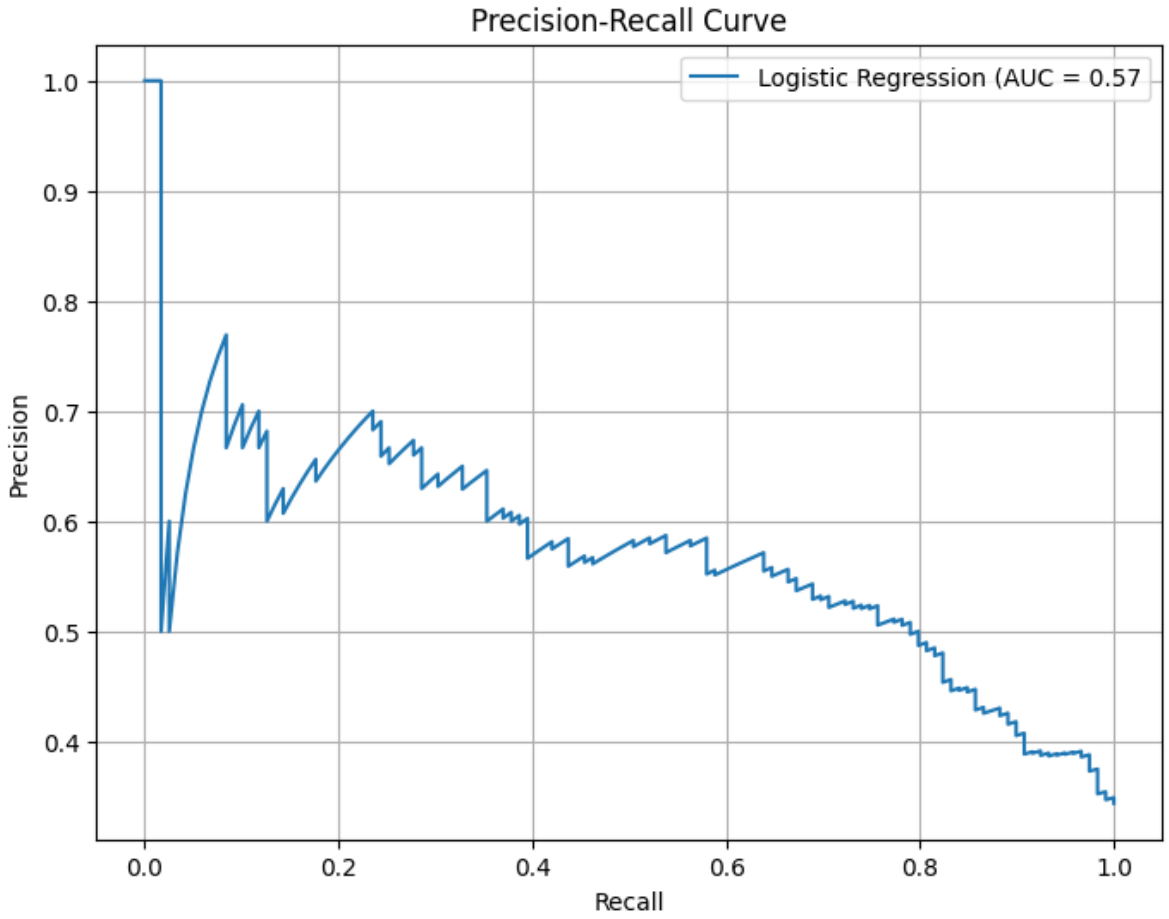
```
0.4369747899159664
```

```
from sklearn.metrics import precision_recall_curve,auc,roc_curve,roc_auc_score
```

```
precisions,recalls,thresholds =  precision_recall_curve(y_train,y_proba)
```

```
auc_pr = auc(recalls,precisions)
```

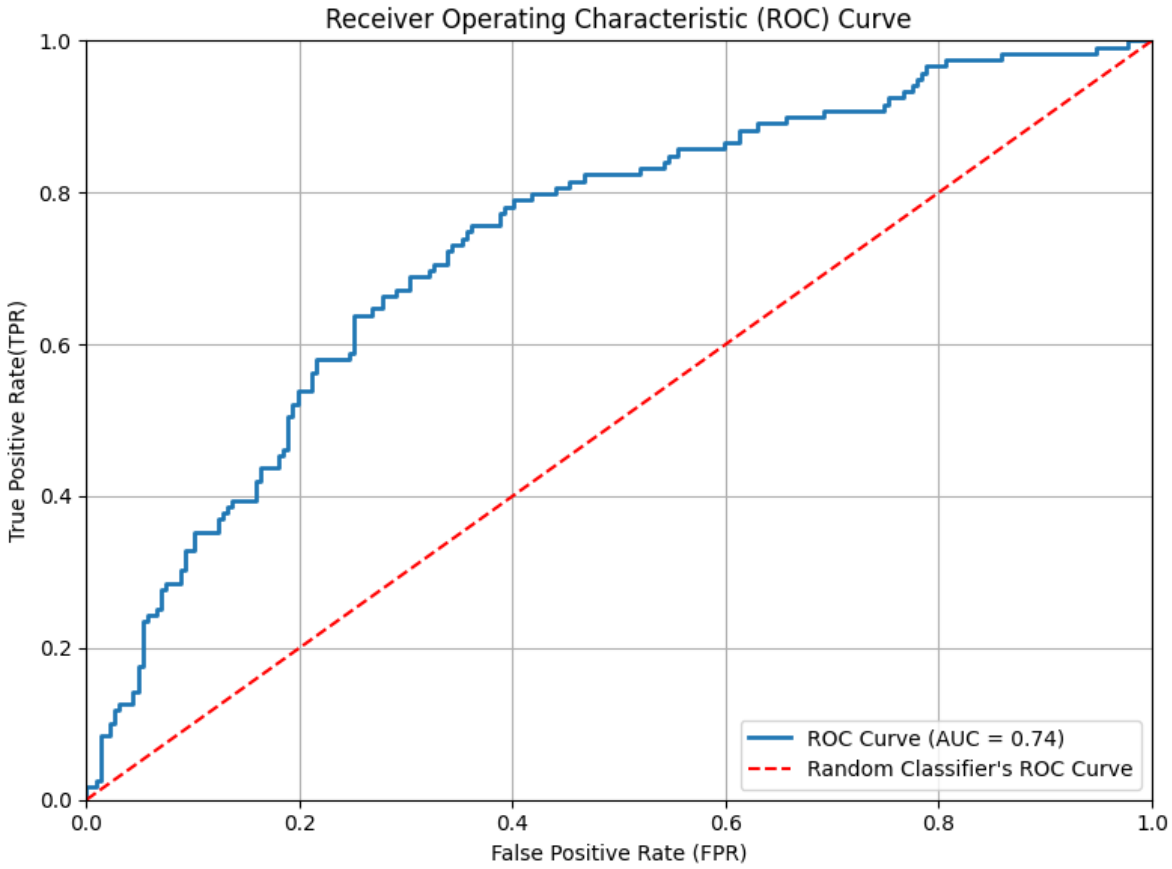
```
fig,ax = plt.subplots(figsize = (8,6))
ax.plot(recalls,precisions,label = f"Logistic Regression (AUC = {auc_pr:.2f}")
ax.set_xlabel("Recall")
ax.set_ylabel("Precision")
ax.set_title("Precision-Recall Curve")
plt.legend(loc = "best")
plt.grid()
plt.show()
```




```
fpr,tpr,thresholds = roc_curve(y_train,y_proba)
```

```
roc_auc = auc(fpr,tpr)
```

```
fig,ax = plt.subplots(figsize =(8,6) )
ax.plot(fpr,tpr,linewidth =2,label = f"ROC Curve (AUC = {roc_auc:.2f})")
ax.plot([0,1],[0,1],"r--", label = "Random Classifier's ROC Curve")
ax.set(
    title = "Receiver Operating Characteristic (ROC) Curve",
    xlabel = "False Positive Rate (FPR)",
    ylabel = "True Positive Rate(TPR)"
)
plt.xlim(0,1)
plt.ylim(0,1)
plt.legend(loc = "lower right")
plt.grid()
plt.tight_layout()
plt.show()
```



```
roc_auc_score(y_train,y_proba)
```

```
0.7365342612816052
```

```
roc_auc_score(y_train,y_scores)
```

```
0.7365342612816052
```

Multiclass Classification

```
beans = pd.read_excel(r"C:\Users\KAsab\Desktop\PYTHON\Dry_Bean_Dataset.xlsx")
```

```
beans.head()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	E
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	1
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	1
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	1
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	1
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	1

```
beans.head()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	E
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	1
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	1
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	1
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	1
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	1

```
np.unique(beans["Class"])
```

```
array(['BARBUNYA', 'BOMBAY', 'CALI', 'DERMASON', 'HOROZ', 'SEKER', 'SIRA'],
      dtype=object)
```

```
beans["Class"].unique()
```

```
array(['SEKER', 'BARBUNYA', 'BOMBAY', 'CALI', 'HOROZ', 'SIRA', 'DERMASON'],
      dtype=object)
```

```
beans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13611 entries, 0 to 13610
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area             13611 non-null  int64  
 1   Perimeter        13611 non-null  float64
 2   MajorAxisLength  13611 non-null  float64
 3   MinorAxisLength  13611 non-null  float64
 4   AspectRation     13611 non-null  float64
 5   Eccentricity     13611 non-null  float64
 6   ConvexArea       13611 non-null  int64  
 7   EquivDiameter    13611 non-null  float64
 8   Extent           13611 non-null  float64
 9   Solidity         13611 non-null  float64
 10  roundness        13611 non-null  float64
 11  Compactness      13611 non-null  float64
 12  ShapeFactor1     13611 non-null  float64
 13  ShapeFactor2     13611 non-null  float64
 14  ShapeFactor3     13611 non-null  float64
 15  ShapeFactor4     13611 non-null  float64
 16  Class            13611 non-null  object 
dtypes: float64(14), int64(2), object(1)
memory usage: 1.8+ MB
```

```
beans.shape
```

```
(13611, 17)
```

```
from sklearn.svm import SVC
from sklearn.multiclass import OneVsRestClassifier
```

```
ovr_clf = make_pipeline(StandardScaler(),OneVsRestClassifier(SVC(random_state = 42)))
```

```
y = beans["Class"]
X = beans.drop(columns = ["Class"])
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 42)
```

```
ovr_clf.fit(X_train,y_train)
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('onevsrestclassifier',
                 OneVsRestClassifier(estimator=SVC(random_state=42)))])
```

```
ovr_svc = Pipeline([
    ("scaler",MinMaxScaler()),
    ("ovr_svm",OneVsRestClassifier(SVC(kernel = "rbf",random_state = 42)))
])
```

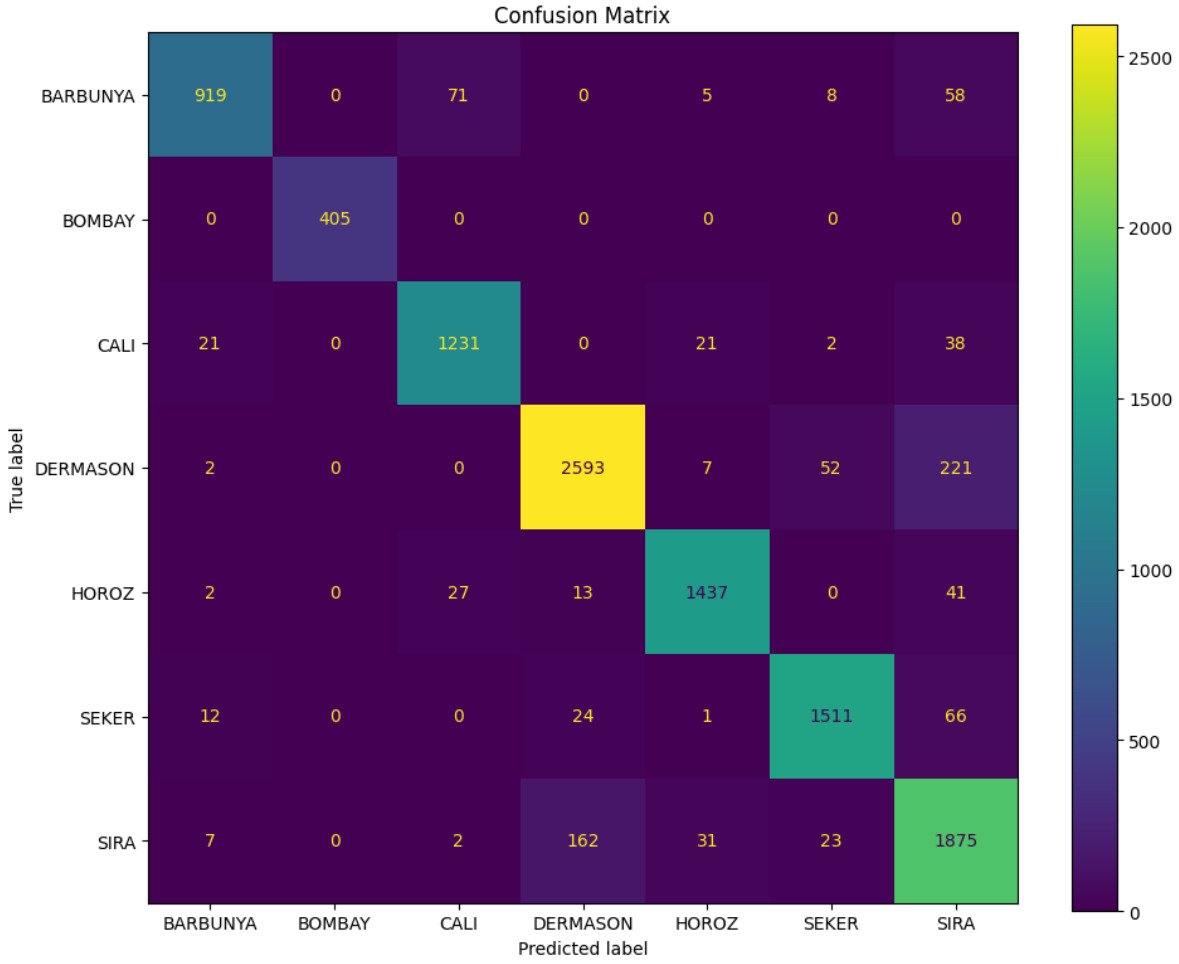
```
ovr_svc.fit(X_train,y_train)
```

```
Pipeline(steps=[('scaler', MinMaxScaler()),
                ('ovr_svm',
                 OneVsRestClassifier(estimator=SVC(random_state=42)))])
```

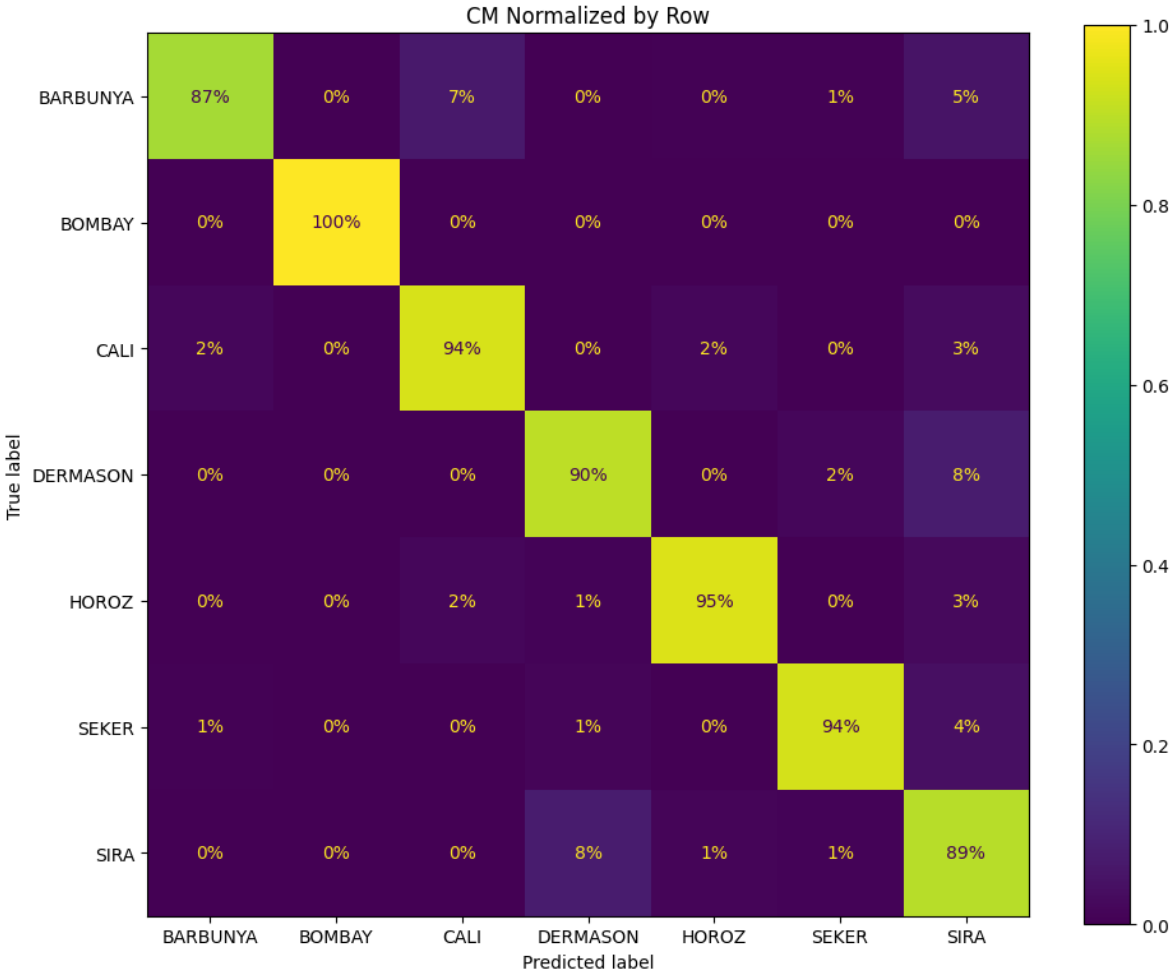
Error Analysis

```
y_trained_pred = cross_val_predict(ovr_svc,X_train,y_train,cv = 3)
```

```
fig,ax = plt.subplots(figsize  = (10,8))
ConfusionMatrixDisplay.from_predictions(y_train,y_trained_pred,ax = ax)
ax.set_title("Confusion Matrix")
plt.tight_layout()
plt.show()
```

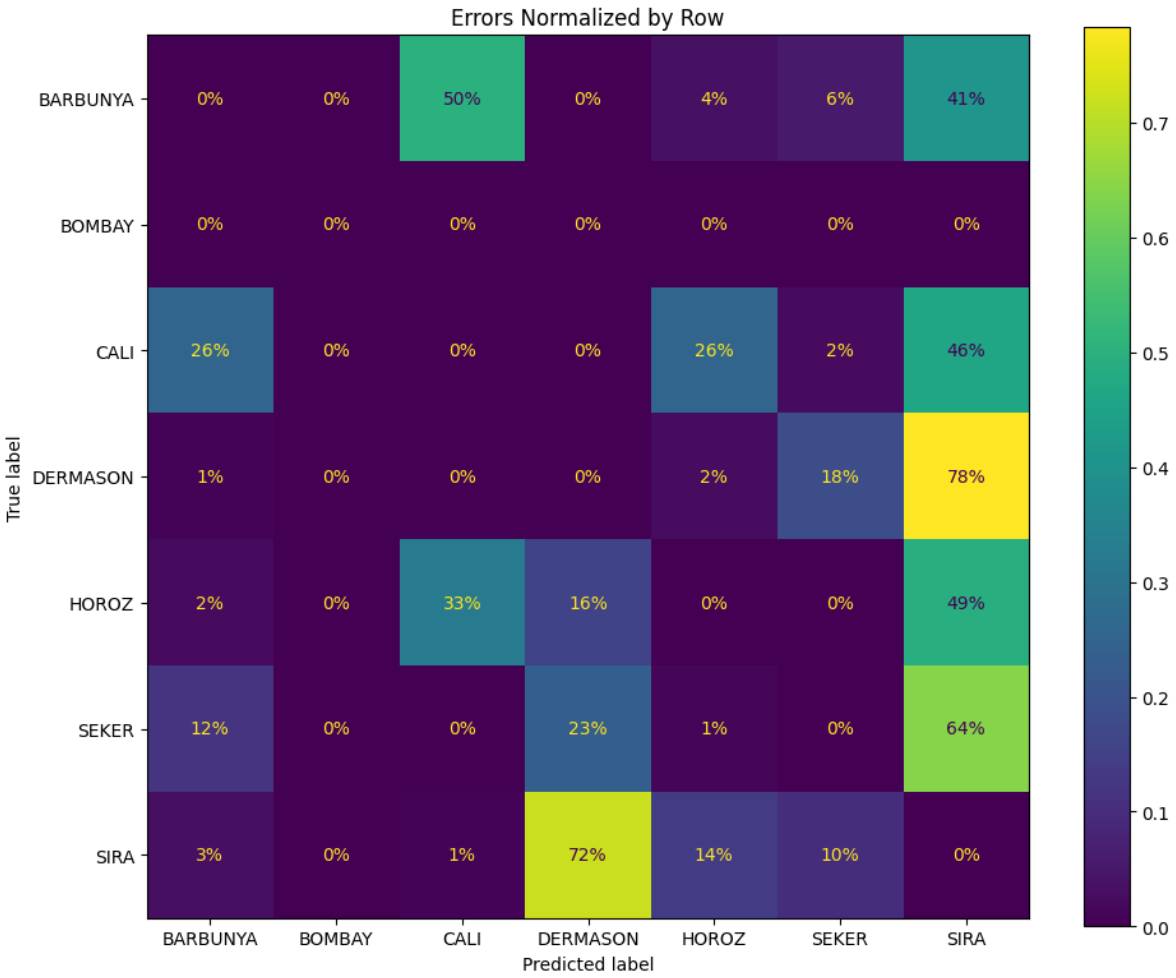



```
fig,ax = plt.subplots(figsize = (10,8))
ConfusionMatrixDisplay.from_predictions(y_train,y_trained_pred,ax = ax,normalize = "true",va
ax.set_title("CM Normalized by Row")
plt.tight_layout()
plt.show()
```



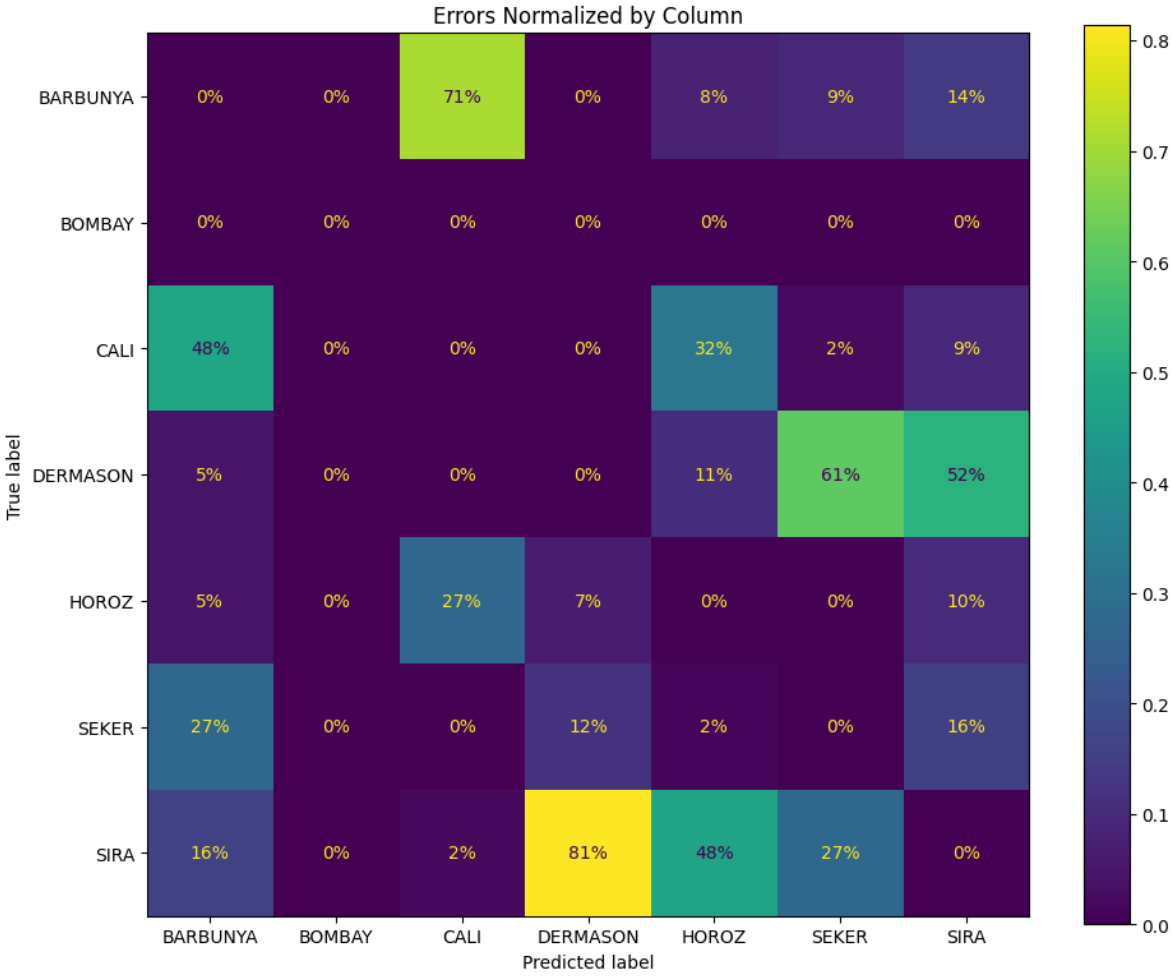

```
sample_weight = (y_trained_pred  != y_train)
```

```
fig,ax = plt.subplots(figsize = (10,8))
ConfusionMatrixDisplay.from_predictions(y_train,y_trained_pred,normalize = "true",
                                        ax = ax,values_format = ".0%",sample_weight = sample_
ax.set_title("Errors Normalized by Row")
plt.tight_layout()
plt.show()
```



78% of the errors made on `DERMASON` were misclassifications as `SIRA`

```
fig,ax = plt.subplots(figsize = (10,8))
ConfusionMatrixDisplay.from_predictions(
    y_train,y_trained_pred,
    sample_weight = sample_weight,
    ax = ax,
    normalize = 'pred',
    values_format = ".0%"
)
ax.set_title("Errors Normalized by Column")
plt.tight_layout()
plt.show()
```


Errors Normalized by Column
BARBUNYA
BOMBAY
CALI
DERMASON
HOROZ
SEKER
SIRA
True label
Predicted label
0% 0% 71% 0% 8% 9% 14%
0% 0% 0% 0% 0% 0% 0%
48% 0% 0% 0% 32% 2% 9%
5% 0% 0% 0% 11% 61% 52%
5% 0% 27% 7% 0% 0% 10%
27% 0% 0% 12% 2% 0% 16%
16% 0% 2% 81% 48% 27% 0%
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0