Métricas de Rendimiento y Ajuste de Hiperparámetros

¿Qué aprenderemos hoy?

- Métricas de Clasificación
- Curva ROC
- Datos Desbalanceados

Métricas de Rendimiento

Matriz de Confusión

In [ ]:
```python
from IPython.display import Image
Image(filename=r'Imagenes_Clase_11/9_6.png', width=400)
```

In [ ]:
```python
Image(filename=r'Imagenes_Clase_11/Confusion_matrix.png', width=400)
```

In [ ]:
```python
from sklearn.metrics import confusion_matrix

y_true = [1, 0, 1, 1, 0, 1, 1, 0, 0, 0]
y_pred = [0, 0, 1, 0, 1, 1, 1, 0, 0, 0]

confusion_matrix(y_true, y_pred)
```

In [ ]:
```python
tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
(tn, fp, fn, tp)
```

In [ ]:
```python
from sklearn.metrics import confusion_matrix

y_true = [1, 2, 1, 1, 0, 1, 1, 0, 2, 0]
y_pred = [0, 0, 1, 0, 1, 1, 1, 0, 2, 0]

confusion_matrix(y_true, y_pred)
```

In [ ]:
```python
import pandas as pd

df = pd.read_csv('https://archive.ics.uci.edu/ml/'
                 'machine-learning-databases'
                 '/breast-cancer-wisconsin/wdbc.data', header=None)


from sklearn.preprocessing import LabelEncoder

X = df.loc[:, 2:].values
y = df.loc[:, 1].values

le = LabelEncoder()
y = le.fit_transform(y)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.20,stratify=y,random_state=1)
print(len(X_train))
print(len(X_test))
```

In [7]:
```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline

from sklearn.svm import SVC

pipe_svc = make_pipeline(StandardScaler(),SVC(random_state=1))
```

In [ ]:
```python
from sklearn.metrics import confusion_matrix

pipe_svc.fit(X_train, y_train)
y_train_pred=pipe_svc.predict(X_train)
y_test_pred = pipe_svc.predict(X_test)


confmat_train = confusion_matrix(y_true=y_train, y_pred=y_train_pred)
print('Matriz de confusión sobre X_train:')
print(confmat_train)


confmat_test = confusion_matrix(y_true=y_test, y_pred=y_test_pred)
print('Matriz de confusión sobre X_test:')
print(confmat_test)
```

In [ ]:
```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(2.5, 2.5),dpi=150)
ax.matshow(confmat_test, cmap=plt.cm.Blues, alpha=0.3)
for i in range(confmat_test.shape[0]):
    for j in range(confmat_test.shape[1]):
        ax.text(x=j, y=i, s=confmat_test[i, j], va='center', ha='center')

plt.xlabel('Etiqueta Predicha')
plt.ylabel('Etiqueta Real')

plt.tight_layout()
#plt.savefig('images/06_09.png', dpi=300)
plt.show()
```

Métricas para evaluar Modelos de Clasificación

Error:

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} \quad (1)$$

Exactitud:

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR \quad (2)$$

Tasa de Verdaderos Positivos:

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \quad (3)$$

Tasa de Falsos Positivos:

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \quad (4)$$

Precisión:

$$PRE = \frac{TP}{TP + FP} \quad (5)$$

Recall:

$$REC = TPR = \frac{TP}{FN + TP} \quad (6)$$

A continuación definiremos:

$$AP = TP + FN, AN = FP + TN \quad (7)$$

$$PP = TP + FP, PN = FN + TN \quad (8)$$

F1-score:

$$F1 = \frac{2 \times TP}{AP + PP} \quad (9)$$

Coeficiente de Correlación de Matthews:

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{PP \times PN \times AP \times AN}} \quad (10)$$

Coeficiente de Kappa de Cohen:

$$K = \frac{P_o - P_e}{1 - P_e} \quad (11)$$

donde,

$$P_e = P_{pos} + P_{neg} \quad (12)$$

$$P_{pos} = \frac{AP}{N} \times \frac{PP}{N} \quad (13)$$

$$P_{neg} = \frac{AN}{N} \times \frac{PN}{N} \quad (14)$$

P_o es la exactitud alcanza por el modelo y N es el total de datos presentados en la matriz de confusión.

In [ ]:
```python
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import matthews_corrcoef,cohen_kappa_score

print('Accuracy (ec. 2): %.3f' % accuracy_score(y_true=y_test, y_pred=y_test_pred))
print('Precision (ec. 5): %.3f' % precision_score(y_true=y_test, y_pred=y_test_pred))
print('Recall (ec. 6): %.3f' % recall_score(y_true=y_test, y_pred=y_test_pred))
print('F1 (ec. 9): %.3f' % f1_score(y_true=y_test, y_pred=y_test_pred))
print('MCC (ec. 10): %.3f' % matthews_corrcoef(y_true=y_test, y_pred=y_test_pred))
print('Kappa (ec. 11): %.3f' % cohen_kappa_score(y1=y_test, y2=y_test_pred))
```

- Documentación

Curva ROC

In [ ]:
```python
from sklearn.metrics import roc_curve, auc
#from scipy import interp
import numpy as np
from sklearn.model_selection import StratifiedKFold
from numpy import interp

pipe_lr = make_pipeline(StandardScaler(),PCA(n_components=2),LogisticRegression(penalty='l2',random_state=1,C=100.0))

X_train2 = X_train[:, [4, 14]]

cv = list(StratifiedKFold(n_splits=5).split(X_train, y_train))

fig = plt.figure(figsize=(7, 5))

mean_tpr = 0.0
mean_fpr = np.linspace(0, 1, 100)
all_tpr = []

for i, (train, test) in enumerate(cv):
    probas = pipe_lr.fit(X_train2[train],y_train[train]).predict_proba(X_train2[test])

    fpr, tpr, thresholds = roc_curve(y_train[test],probas[:, 1],pos_label=1)
    mean_tpr += interp(mean_fpr, fpr, tpr)
    mean_tpr[0] = 0.0
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr,tpr,label='ROC fold %d (area = %0.2f)'% (i+1, roc_auc))

plt.plot([0, 1],[0, 1],linestyle='--',color=(0.6,0.6,0.6),label='random guessing')

mean_tpr /= len(cv)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
plt.plot(mean_fpr, mean_tpr, 'k--',label='mean ROC (area = %0.2f)' % mean_auc,lw=2)
plt.plot([0, 0, 1],[0, 1, 1],linestyle=':',color='black',label='perfect performance')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.legend(loc="lower right")

plt.tight_layout()
# plt.savefig('images/06_10.png', dpi=300)
plt.show()
```

Tratar con Datos Desbalanceados

In [ ]:
```python
X[y==0].shape
```

In [32]:
```python
X_imb = np.vstack((X[y == 0], X[y == 1][:15]))
y_imb = np.hstack((y[y == 0], y[y == 1][:15]))
```

In [ ]:
```python
X_imb.shape
```

In [ ]:
```python
print('Cantidad de datos en las clases:', np.bincount(y_imb))
```

In [ ]:
```python
y_pred = np.zeros(y_imb.shape[0])
np.mean(y_pred == y_imb) * 100
```

In [ ]:
```python
from sklearn.utils import resample

print('Número de muestras de clase 1 antes:', X_imb[y_imb == 1].shape[0])
```

In [ ]:
```python
X_upsampled, y_upsampled = resample(X_imb[y_imb == 1],y_imb[y_imb == 1],replace=True,
                                    n_samples=X_imb[y_imb == 0].shape[0],random_state=123)

print('Número de muestras de clase 1 después:', X_upsampled.shape[0])
```

In [50]:
```python
X_bal = np.vstack((X[y == 0], X_upsampled))
y_bal = np.hstack((y[y == 0], y_upsampled))
```

In [ ]:
```python
y_pred = np.zeros(y_bal.shape[0])
np.mean(y_pred == y_bal) * 100
```

Actividad

1. Considere los datos Pokemon_DB asociados a la Ayundantía 2 después de la limpieza sugerida en la ayudantía.
2. Utilice los datos de las dos clases mayoritarias y examine el balance de los mismos.
3. Proponga una metodología para construir una base de datos nueva Pokemon_DB_balanceada a partir de la que tiene, pero que tenga mejor balance de clases. Describa las metodologías usa

4. Entrene un modelo predictor de las clases con ambas bases de datos y compare los resultados utilizando la matriz de confusión.
5. Calcule algunas de las métricas introducidas y dibuje las curvas ROC.