{desafío}
latam_

Análisis discriminante _

Sesión Presencial 2

Itinerario

Activación de conceptos

Desarrollo Desafío

Panel de discusión

{desafío}
latam_

Activación de conceptos

¿Qué métricas podemos utilizar en un problema de clasificación?

- MSE, RMSE, MAE
- ROC, F1, Accuracy, Precision, Recall
- Ambas son útiles

¿Cómo implementarías Naïve Bayes con más de 2 clases con scikit-learn?

- Recodificamos k-1 categorías en variables binarias
- Con `sklearn.naive_bayes.MultinomialNB`
- Con `sklearn.discriminant_analysis.MultinomialDiscriminantAnalysis`

¿Qué método de `scikit-learn.naive_bayes` ocupamos si todos nuestros atributos son continuos?

- `sklearn.naive_bayes.MultinomialNB`
- `sklearn.naive_bayes.GaussianNB`
- `sklearn.naive_bayes.BernoulliNB`

Intuición

Deseamos encontrar una dimensión que maximice la distancia entre $N_{\mathbb{Y}}$ clases.
LDA es un algoritmo generativo Pr(y|x) → Pr(x,y) dado $\Pr(y|x) \rightsquigarrow \Pr(x,y)$ dado $x \in \mathbf{X}^{\mathbb{R}}$, e $y \in \mathbb{Y}$.

Objetivo:
Identificar la pertenencia de una observación a una clase específica.

Diferencia con Naïve Bayes
Asumimos que los datos condicionales a la clase, se comportan de forma normal

Multivariada Normal

Tenemos:

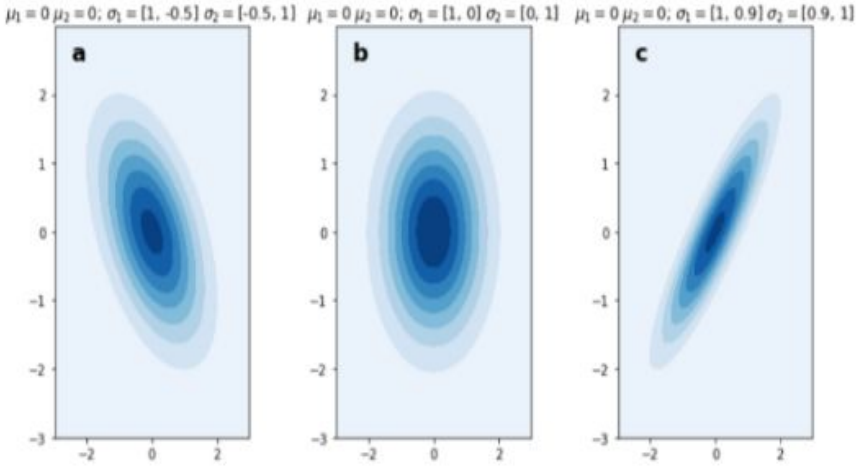
$$\text{MultivarNorm}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{2\pi^{N/2}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(x-\boldsymbol{\mu})^{\top} \right]$$

$$y \in \mathbb{Y}, \text{e}\, \boldsymbol{\Sigma}$$

$\boldsymbol{\mu}$ un vector de medias $y \in \mathbb{Y}, \text{e}\, \boldsymbol{\Sigma}$

es una matriz de covarianza (fija) entre los atributos.

```
In [2]: plt.figure(figsize=(10, 5));afx.plot_mvn();plt.tight_layout()
```

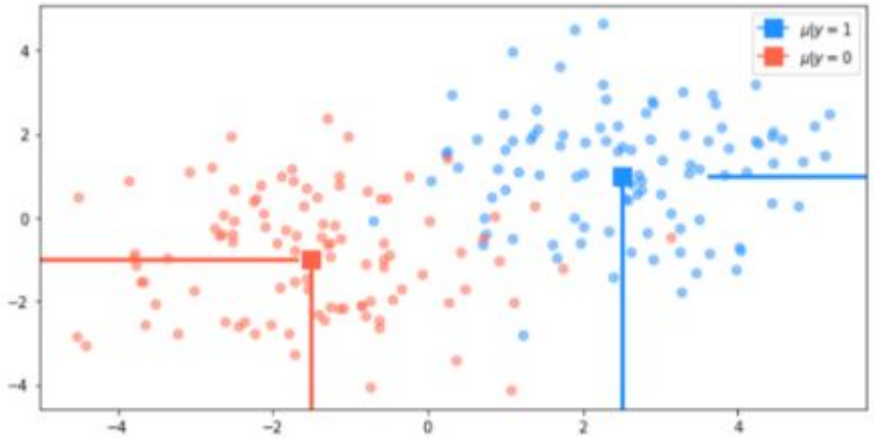


{desafío} latam_

Implementación

1. Extraer vectores μ para cada clase

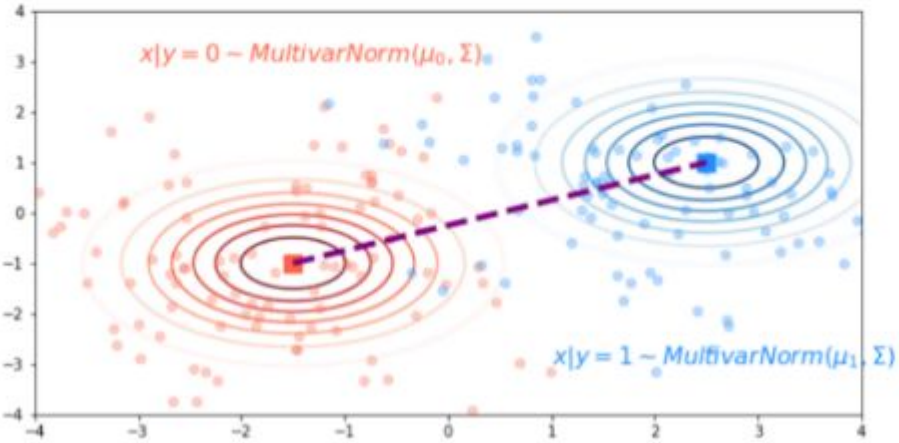
```
In [3]: plt.figure(figsize=(10, 5));afx.plot_features()
```



2. Estimar las densidades multivariadas para cada clase

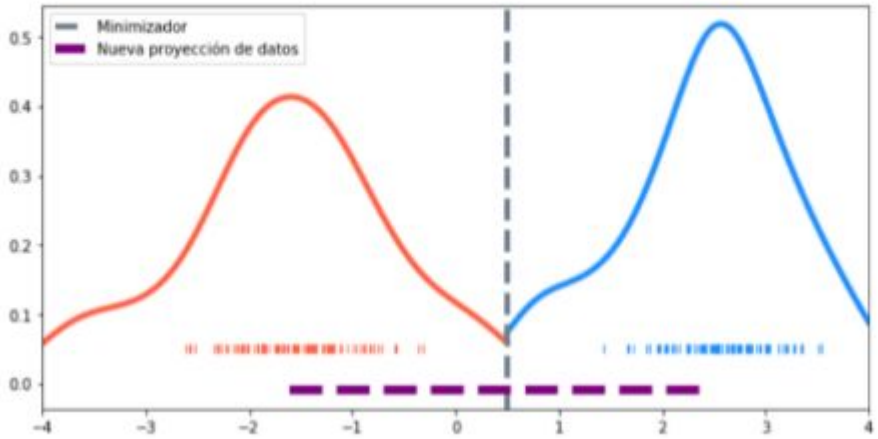
$$x|y \in \mathbb{Y} \sim \text{MultivarNorm}(\boldsymbol{\mu}_i, \Sigma)$$

```
In [4]: plt.figure(figsize=(10, 5));afx.plot_lda()
```



3. En base a las densidades inferidas, seleccionar un discriminante lineal

```
In [5]: plt.figure(figsize=(10, 5));afx.plot_densities()
```



Implementación con sklearn

Preparación del ambiente

In [6]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

Preprocesamiento

In [7]:
```
# botamos la columna de index
df = pd.read_csv('iris.csv').drop(columns='Unnamed: 0')
# guardamos las etiquetas de cada clase para posterior uso.
target_label = df['Species'].unique()
df['Species'] = LabelEncoder().fit_transform(df['Species'])
X_train_mat, X_test_mat, y_train_vec, y_test_vec = train_test_split(df.loc[:, 'Sepal.Leng
th':'Petal.Width'],df['Species'],test_size=.33, random_state=11238)
```

Instanciamos la clase

```
In [8]: lda_model = LinearDiscriminantAnalysis()
        lda_model.fit(X_train_mat, y_train_vec)

Out[8]: LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=N
        one,
                                   solver='svd', store_covariance=False, tol=0.0001)
```

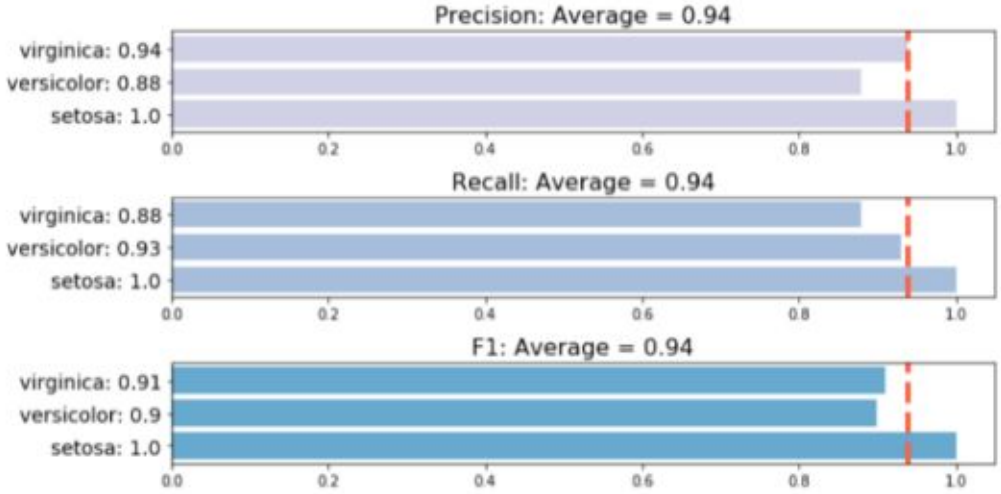
Algunos estimados relevantes

In [9]:
```
print("Probabilidades a priori\n", lda_model.priors_, "\n")
print("Medias de atributos en cada clase\n", lda_model.means_.round(2), "\n")
print("Varianza explicada por cada discriminante\n", lda_model.explained_variance_ratio_.
round(2), "\n")
```

```
Probabilidades a priori
 [0.32 0.35 0.33]

Medias de atributos en cada clase
 [[5.08 3.54 1.47 0.27]
 [5.96 2.8  4.32 1.35]
 [6.57 2.96 5.52 2.05]]

Varianza explicada por cada discriminante
 [0.99 0.01]
```

Evaluación de predicciones

```
In [10]: lda_class_pred = lda_model.predict(X_test_mat)
         lda_class_pred[:10]

Out[10]: array([2, 1, 0, 2, 2, 1, 0, 1, 0, 2])

In [11]: plt.figure(figsize=(10, 5));afx.plot_class_report(y_test_vec, lda_class_pred, classes_lab
         els=target_label)
```



{desafío} latam_

Quadratic Discriminant Analysis

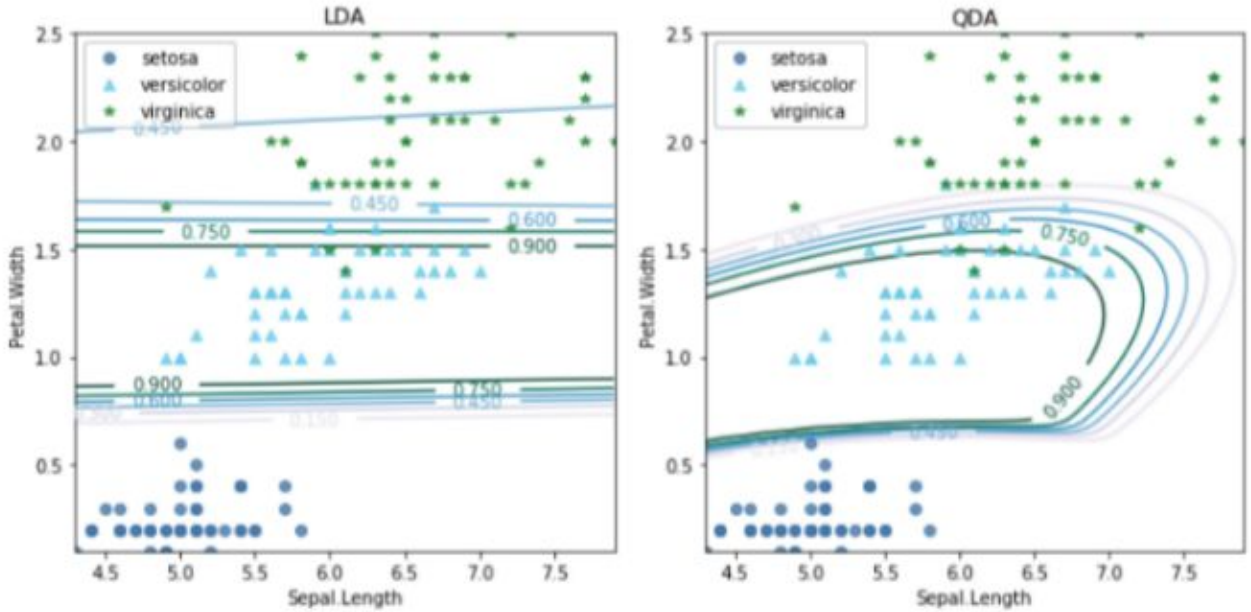
Intuición

- LDA → Discriminantes lineales
- QDA supera ésto al elevar cada atributo al cuadrado
- También asume Σ_k

```
In [12]: from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
         qda_model = QuadraticDiscriminantAnalysis().fit(X_train_mat, y_train_vec)
         qda_class_pred = qda_model.predict(X_test_mat)
```

Comparación

```
In [14]: plt.figure(figsize=(10, 5)); lda_qda_comparison(); plt.tight_layout()
```



/* Desafío */

Panel de discusión

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com