{desafío}
latam_

Clasificación_

Sesión Presencial 2

Activación de Conceptos

- En la unidad anterior aprendimos sobre la regresión logística desde la econometría.
- ¡Pongamos a prueba nuestros conocimientos!

¿Por qué no debemos fijarnos en el R^2 para los modelos logísticos?

1. Porque son aproximaciones al método de mínimos cuadrados.

- Porque el criterio de optimización es la maximización de la verosimilitud.
- Porque no hay varianza en nuestro vector objetivo.

¿Qué permite la función logística inversa?

1. Mapear de log-odds a probabilidad.

- Mapear de log-odds a odds-ratios.
- Mapear de probabilidad a log-odds.

¿Cómo podemos aproximarnos rápidamente a la probabilidad de un log-odd?

1. Convirtiendo el beta con la función logística inversa.

- Dividiendo por 4
- Dividiendo por 2.5

Regresión Logística (Desde Machine Learning)

{desafío}
latam_

Flujo de trabajo

- Para montar un modelo predictivo de clasificación, seguimos los mismos pasos:

Cargamos módulos

In [9]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Estandarizamos y dividimos la muestra

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(StandardScaler().fit_transform(df.loc
[:, 'dist100':'assoc']),df['y'], test_size=.33, random_state=11238)
```

Entrenamos el modelo

In [11]:
```
model = LogisticRegression().fit(X_train, y_train)
```

Solicitamos métricas (clases predichas)

In [12]:
```
yhat_class = model.predict(X_test)
yhat_class[:10]
```

Out[12]:
```
array([0, 1, 1, 0, 1, 1, 1, 1, 1, 1])
```

Solicitamos métricas ($\mathbf{Pr(y^*) \mapsto [0, 1]}$)

In [13]:
```
yhat_pr = model.predict_proba(X_test)[:, 1]
list(map(lambda x: round(x, 2), yhat_pr[:10]))
```

Out[13]:
```
[0.32, 0.67, 0.91, 0.3, 0.6, 0.54, 0.69, 0.65, 0.54, 0.53]
```

Métricas de Clasificación

{desafío}
latam_

Matriz Confusa

- Tabla cruzada entre etiquetas predichas y etiquetas verdaderas

In [14]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, yhat_class))
```

```
[[174 248]
 [120 455]]
```

- De acá surge la **Exactitud, Precision, Recall y F1** del modelo

In [15]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, yhat_class))
```

```
             precision    recall  f1-score   support

          0       0.59      0.41      0.49       422
          1       0.65      0.79      0.71       575

avg / total       0.62      0.63      0.62       997
```

{desafío} latam_

Curva ROC

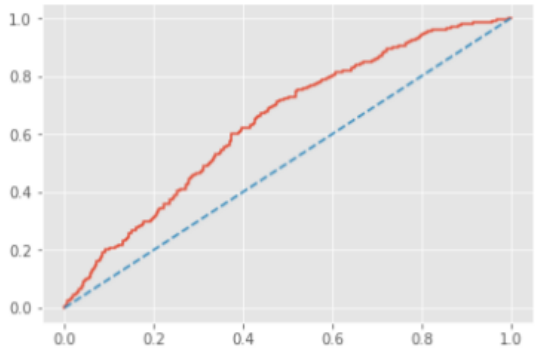
- Cruce entre **Falsos Positivos** y **Verdaderos Positivos**.

In [ ]:
```
from sklearn.metrics import roc_curve
```

In [ ]:
```
false_positive, true_positive, thres = roc_curve(y_test, yhat_pr)
```

In [ ]:
```
plt.plot(false_positive, true_positive)
```

In [16]:
```
plt.plot([0,1], ls='--');
```



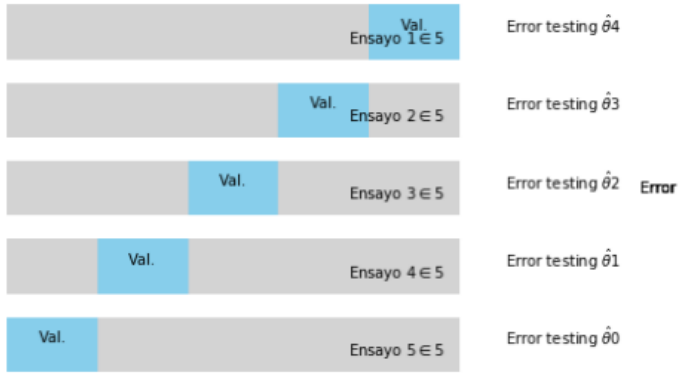
Resampling

{desafío}
latam_

k-Fold Cross Validation

- ¿Qué es mejor que dividir la muestra una vez? Dividir la muestra muchas veces!
- La validación cruzada permite resolver el problema de observaciones insuficientes.

In [17]:
```
fig=plt.figure();ax=fig.add_axes([0,0,1,1]);ax.axis('off');gfx.crossvalidation_schema(ax=
ax, cv_folds=5)
```



{desafío}
latam_

Implementando k-fold Cross Validation

In [ ]:
```
from sklearn.model_selection import cross_val_score
```

In [ ]:
```
m1_cv = cross_val_score(LogisticRegression(), X=df.loc[:, 'dist100':'assoc'], y=df.loc[:,
'y'], cv=3, scoring='f1')
```

In [18]:
```
print("Puntajes Individuales: ", m1_cv)
print("Promedio de cada iteración: ", np.mean(m1_cv))
print("Desviación Estándar de cada iteración: ", np.std(m1_cv))
```

```
Puntajes Individuales:  [0.70499244 0.68932806 0.69954476]
Promedio de cada iteración:  0.6979550879132436
Desviación Estándar de cada iteración:  0.006492993434034528
```

Leave-one-out Cross Validation

- LOOCV es una situación especial donde cada observación es un ensayo.
- En este esquema se separara una observación de la muestra para ser predicha.

Implementación

In [ ]:
```
%%time
from sklearn.model_selection import LeaveOneOut
```

In [19]:
```
m1_loocv = cross_val_score(LogisticRegression(), df.loc[:, 'dist100': 'assoc'], df.loc[:,
'y'], cv=LeaveOneOut(),scoring='accuracy')
```

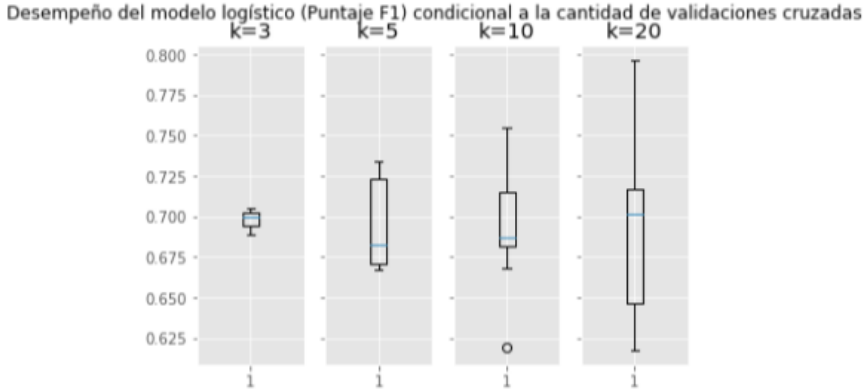
```
CPU times: user 13.9 s, sys: 66.5 ms, total: 14 s
Wall time: 14.1 s
```

In [20]:
```
print(m1_loocv[:20])
print(np.mean(m1_loocv))
print(np.std(m1_loocv))
```

```
[1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.]
0.6139072847682119
0.4868522676097273
```

Comportamiento

In [22]:
```
varying_k_cv()
```



In [ ]:
```

```