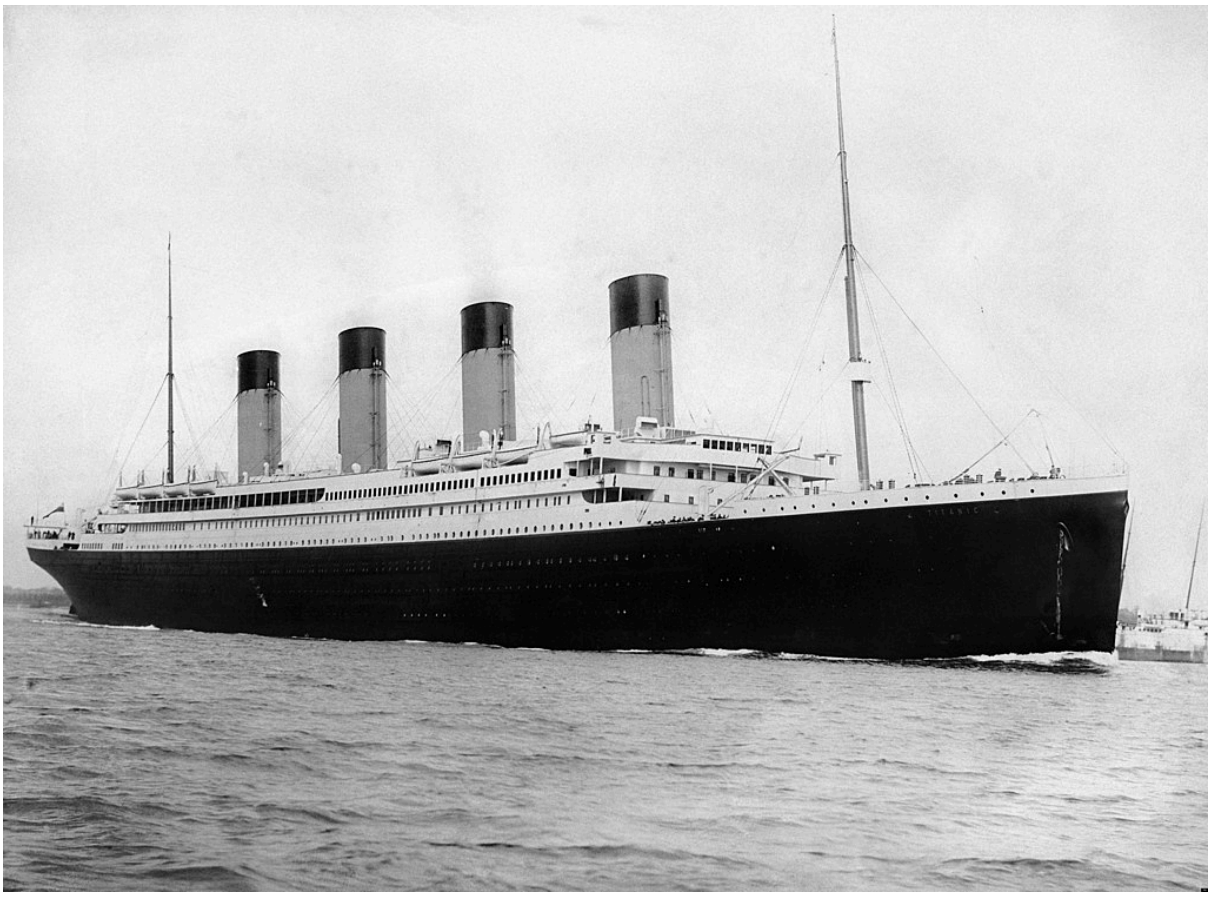
Representación plot de datasets, selección de características y entrenamiento de modelos



Adrián Yared Armas de la Nuez

Contenido

1. Objetivo de la actividad

El objetivo de esta actividad es poner en práctica los conocimientos adquiridos para el preprocesamiento de datos, selección de características y entrenamiento de modelos.

Para ello es necesario seleccionar un Dataset que consideres oportuno, de clasificación o regresión, y distinto a los utilizados en clase.

2.1 Apartados de la actividad

2.1.1 Enunciado 1

Describir el origen y breve explicación del Dataset, así como de cada una de las características.

2.1.2 Resolución

El dataset Titanic proviene de los registros de pasajeros del RMS Titanic, el transatlántico que se hundió en 1912. Contiene información sobre los pasajeros, como su clase de boleto, sexo, edad, número de familiares a bordo, tarifa del pasaje y si sobrevivieron. Las características incluyen: survived (si el pasajero sobrevivió o no), pclass (clase del boleto), sex (sexo), age (edad), sibsp (número de hermanos/cónyuges), parch (número de padres/hijos), fare (tarifa), embarked (puerto de embarque), who (género) y alone (viajaba solo o acompañado). Este conjunto de datos es comúnmente utilizado en el aprendizaje de técnicas de modelado predictivo y análisis exploratorio, ayudando a comprender cómo diversas variables influyeron en la supervivencia de los pasajeros.

2.1.3 Enunciado 2

Procesamiento de datos en el dataset: ajustes en características con datos no informados, conversión de variables categóricas, etc...

2.1.4 Código

Ajustes de los datos para la posterior utilización:

```
# 2. Procesamiento de datos
# Eliminar columnas con demasiados valores faltantes
threshold = 0.4  # Porcentaje mínimo de valores no nulos requeridos
cols_to_keep = data.columns[data.isnull().mean() < (1 - threshold)]
data = data[cols_to_keep]

# Llenar valores faltantes
for col in data.columns:
```

```
    if data[col].dtype == 'object' or data[col].dtype.name ==
'category':
        # Si es categórico, llenamos con la moda
        data[col] = data[col].fillna(data[col].mode()[0])
    else:
        # Si es numérico, llenamos con la media
        data[col] = data[col].fillna(data[col].mean())

# Conversión de variables categóricas
label_encoders = {}
for col in data.select_dtypes(include='object').columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Separar características y la variable objetivo
target = 'survived'
X = data.drop(columns=[target])
y = data[target]

# Escalado de datos
numeric_columns = X.select_dtypes(include=['float64', 'int64']).columns
categorical_columns = X.select_dtypes(include=['object',
'category']).columns

scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X[numeric_columns]),
columns=numeric_columns)
X_categorical = X[categorical_columns]

# Combinar datos escalados y categóricos
X = pd.concat([X_scaled, X_categorical.reset_index(drop=True)], axis=1)
```

2.1.5 Enunciado 3

Utilizar las siguientes herramientas explicadas en clase para la selección de características

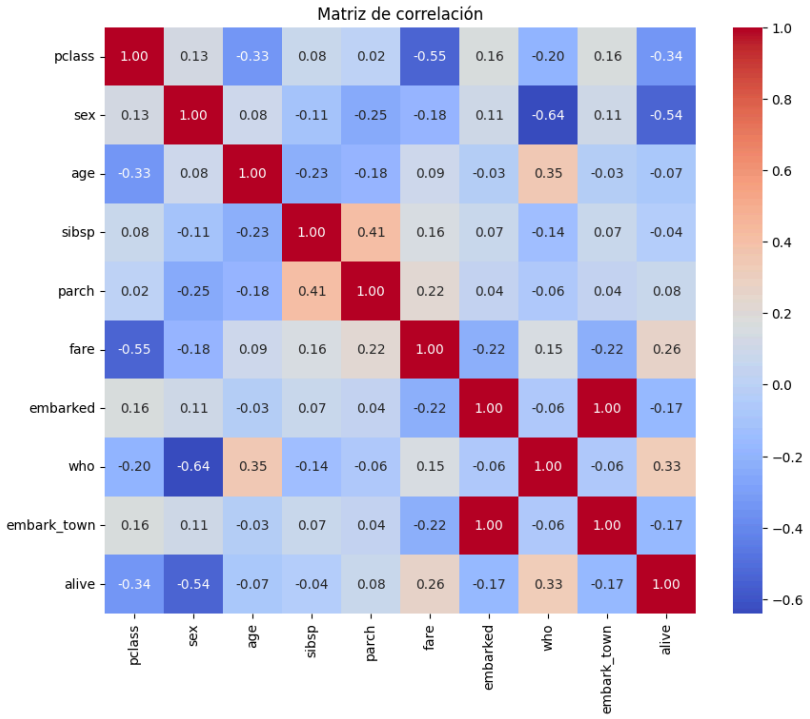
2.1.5.1 Enunciado 3.1

Matriz de gráficos de correlación.

2.1.5.2 Código

```
# 3.1 Matriz de gráficos de correlación
plt.figure(figsize=(10, 8))
numeric_columns = X.select_dtypes(include=['float64', 'int64']).columns
X_numeric = X[numeric_columns]
correlation_matrix = X_numeric.corr()
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Matriz de correlación')
plt.show()
```

2.1.5.3 Resultado



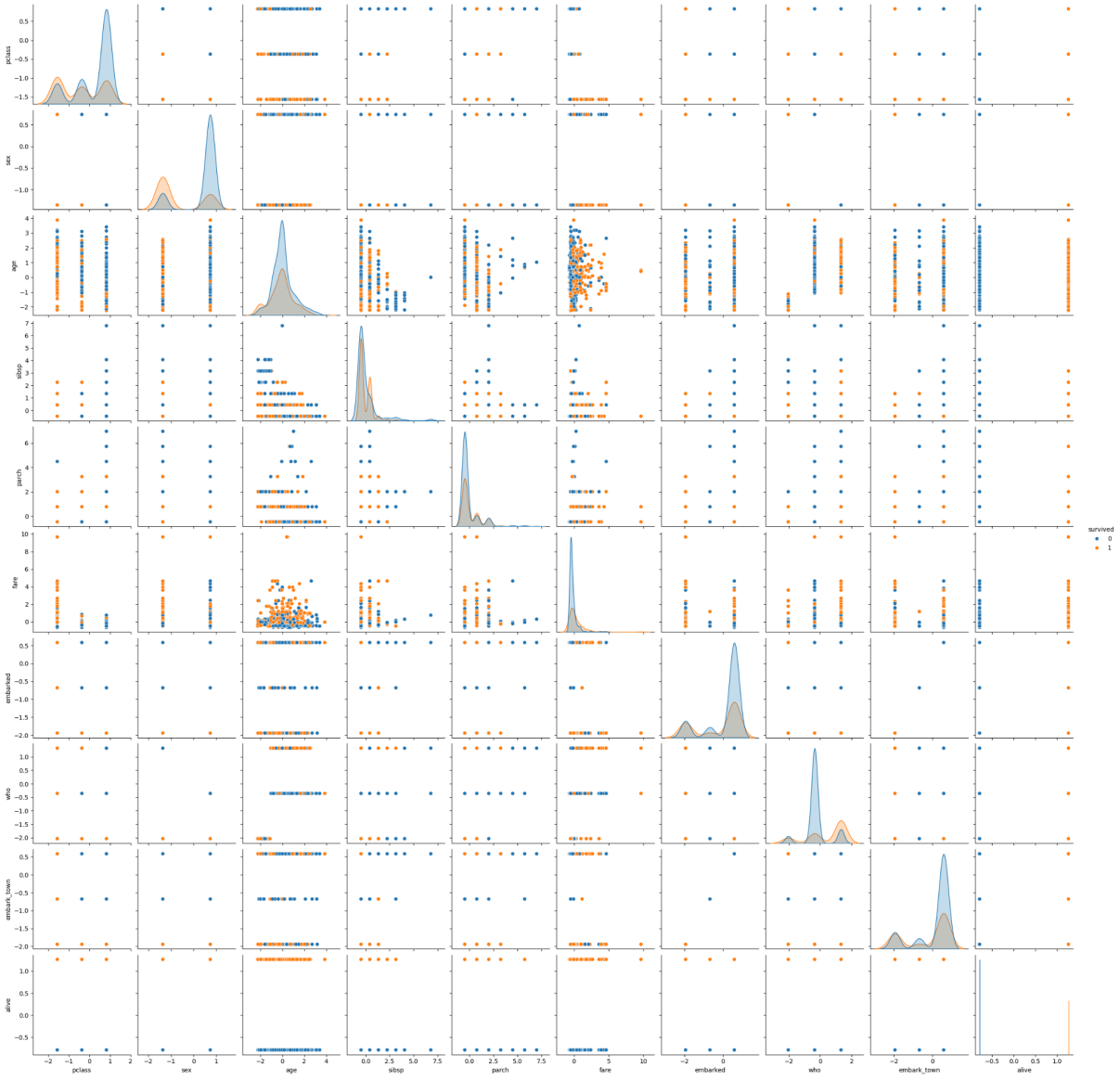
2.1.5.4 Enunciado 3.2

Matriz de gráficos de dispersión.

2.1.5.5 Código

```
# 3.2 Matriz de gráficos de dispersión
print(f'Matriz de Dispersión: ')
sns.pairplot(pd.concat([X_numeric, y], axis=1), diag_kind='kde',
hue=target)
plt.show()
```

2.1.5.6 Resultado



2.1.5.7 Enunciado 3.3

SelectKBest.

2.1.5.8 Código

```
# 3.3 SelectKBest
k = 5  # Seleccionar las 5 mejores características
skb = SelectKBest(score_func=chi2, k=k)
X_selected = skb.fit_transform(abs(X_numeric), y)
selected_features = X_numeric.columns[skb.get_support()]
print(f"Características seleccionadas (SelectKBest):
{selected_features.tolist()}")
```

2.1.5.9 Resultado

```
Características seleccionadas (SelectKBest): ['sex', 'sibsp', 'fare', 'who', 'alive']
```

2.1.6 Enunciado 4

Una pequeña reflexión sobre la elección de las características elegidas.

2.1.7 Resolución

Elegí variables relevantes para predecir la supervivencia en el Titanic. La clase del boleto (Pclass) es importante, ya que los pasajeros de primera clase tuvieron mayores probabilidades de sobrevivir. El sexo también influyó, con una mayor tasa de supervivencia entre las mujeres. La edad es relevante, ya que niños y personas mayores fueron rescatados con más frecuencia. El número de hermanos y/o cónyuges (Sibsp) a bordo podría haber afectado las oportunidades de evacuación, mientras que la tarifa del pasaje (Fare) se relaciona con la clase social y las probabilidades de sobrevivir. Estas características, aunque no exhaustivas, cubren varias dimensiones clave que influyeron en la supervivencia.

2.1.8 Enunciado 5

Con las librerías para NaiveBayes vistas en clase, entrenar el modelo que consideres más adecuado.

2.1.9 Código

Dado que no estaba seguro si era más preciso usar el modelo GaussianNB o BernoulliNB por mi falta de experiencia hasta la fecha, hice una comparativa tanto de estos como el resto de modelos explicados en clase para ser más certero con la elección del modelo. Esto está en el apartado “Todos los modelos” del Notebook. Y cuyos resultados son los siguientes:

```
Comparación de resultados:
              Sin Cross Validation  Con Cross Validation
GaussianNB                1.000000              1.000000
MultinomialNB             0.642458              0.653192
ComplementNB              0.798883              0.805819
BernoulliNB               0.586592              0.616163
CategoricalNB             1.000000              1.000000
```

```python
# Función para entrenar y evaluar el modelo
def train_and_evaluate(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    return accuracy_score(y_test, y_pred)
```

```
# Modelos seleccionados: GaussianNB y BernoulliNB
models = {
    'GaussianNB': GaussianNB(),
    'BernoulliNB': BernoulliNB()
}
```

2.1.9.1 Sin utilizar Cross Validation (Enunciado 5.1)

```
 Evaluar sin Cross Validation
print("Resultados sin Cross Validation:")
results_no_cv = {}
for name, model in models.items():
    try:
        acc = train_and_evaluate(model, X_train, X_test, y_train,
y_test)
        results_no_cv[name] = acc
        print(f"{name}: {acc:.2f}")
    except Exception as e:
        print(f"{name}: Error ({e})")
```

2.1.9.2 Utilizando Cross Validation (Enunciado 5.2)

```
# Evaluar con Cross Validation (CV=10 para mayor robustez)
print("\nResultados con Cross Validation:")
results_cv = {}
for name, model in models.items():
    try:
        scores = cross_val_score(model, X_reduced, y, cv=10,
scoring='accuracy')
        results_cv[name] = scores.mean()
        print(f"{name}: {scores.mean():.2f}")
    except Exception as e:
        print(f"{name}: Error ({e})")
```

2.1.10 Enunciado 6

Obtener una conclusión sobre los resultados obtenidos en la predicción y evaluación al utilizar o no Cross Validation.

2.1.11 Código

```
# Comparación de resultados
```

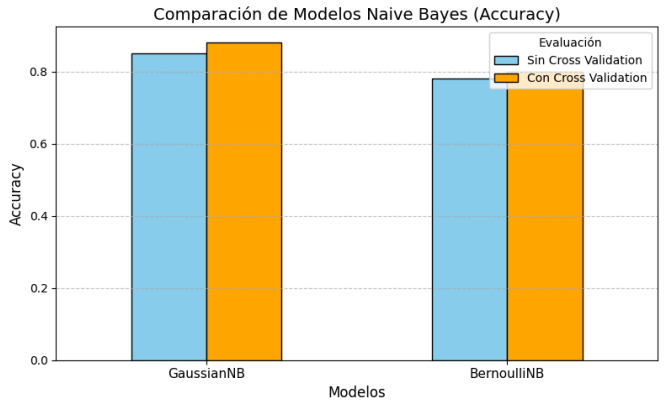
```
print("\nComparación de resultados:")
comparison = pd.DataFrame({
    "Sin Cross Validation": results_no_cv,
    "Con Cross Validation": results_cv
})
print(comparison)
```

2.1.12 Resultado

```
Resultados sin Cross Validation:
GaussianNB: 0.80
BernoulliNB: 0.79

Resultados con Cross Validation:
GaussianNB: 0.81
BernoulliNB: 0.79

Comparación de resultados:
             Sin Cross Validation  Con Cross Validation
GaussianNB               0.798883              0.808052
BernoulliNB              0.787709              0.793508
```

2.1.13 Breve conclusión de los resultados

Los resultados muestran que, en general, la cross validation mejora ligeramente la precisión de los modelos comparado con el uso de un único conjunto de prueba. Para GaussianNB, la precisión aumenta de 0.7989 a 0.8081, mientras que para BernoulliNB, la mejora es de 0.7877 a 0.7935. Aunque las diferencias son pequeñas, cross validation ofrece una estimación más robusta y generalizada del rendimiento. Esto sugiere que el uso de Cross Validation es preferible para evaluar modelos en tareas con alta variabilidad en los datos, como en este caso.



2.1.14 Enunciado (Enunciado 7)

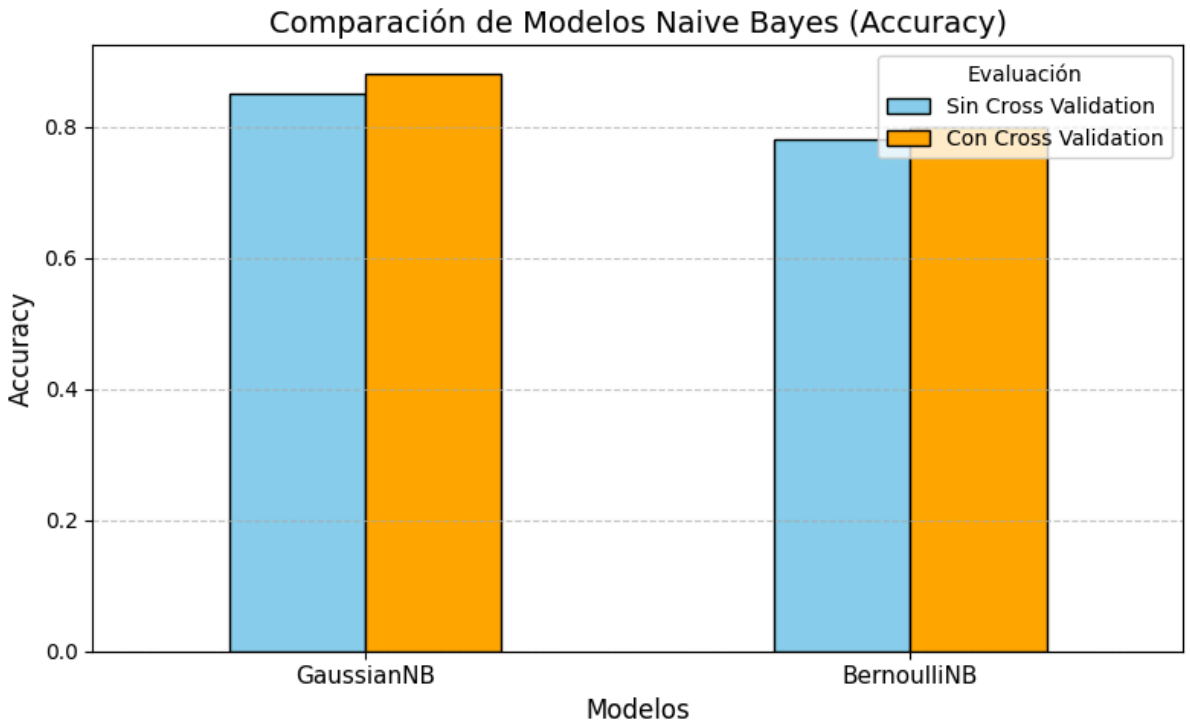
Además de las herramientas indicadas anteriormente, se valorará la utilización de alguna otra herramienta o técnica no vista en el curso para la selección de las características.

2.1.15 Código

Mi aportación es una comparativa de precisiones a través de un gráfico de barras con el fin de hacerlo más visual y aportar una conclusión más clara.

```
import matplotlib.pyplot as plt
import pandas as pd

# Datos simulados de resultados (puedes reemplazarlos con tus
resultados reales)
results_no_cv = {'GaussianNB': 0.85, 'BernoulliNB': 0.78}
results_cv = {'GaussianNB': 0.88, 'BernoulliNB': 0.80}

# Crear un DataFrame para los resultados
comparison = pd.DataFrame({
    "Sin Cross Validation": results_no_cv,
    "Con Cross Validation": results_cv
})

# Gráfico de barras para comparar los resultados
fig, ax = plt.subplots(figsize=(8, 5))
comparison.plot(kind='bar', ax=ax, color=['skyblue', 'orange'],
edgecolor='black')
ax.set_title('Comparación de Modelos Naive Bayes (Accuracy)',
fontsize=14)
ax.set_ylabel('Accuracy', fontsize=12)
ax.set_xlabel('Modelos', fontsize=12)
ax.set_xticklabels(comparison.index, rotation=0, fontsize=11)
ax.legend(title='Evaluación', fontsize=10)
ax.grid(axis='y', linestyle='--', alpha=0.7)

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

2.1.16 Resultado



3. Bibliografía

Apuntes de la asignatura:
https://www3.gobiernodecanarias.org/medusa/eforma/campus/pluginfile.php/8849167/mod_resource/content/50/UT3%20-%20Algoritmos%20y%20herramientas%20para%20el%20aprendizaje%28I%29_v2.pdf

Matriz de dispersión:
https://interactivechaos.com/es/manual/tutorial-de-matplotlib/graficos-de-dispersion

4. Github y Colab