Universidad Nacional Autónoma de México.
IIMAS

RECONOCIMIENTO DE PATRONES

Semestre 2025-2.

Dra. Mar ıa Elena Mart ınez Perez

M. en C. Miguel Angel Veloz Lucas

Practica 3

Integrantes:

- Villalón Pineda Luis Enrique .

I. Objetivos

- Implementar diferentes métodos de selección de características
- Comparar resultados entre distintos métodos de selección de características.
- Analizar el impacto de la reducción dimensional.

II. Introducción

// La selección de características es fundamental en el campo del reconocimiento de patrones y el aprendizaje automático. Su propósito radica en identificar las variables o atributos más informativos dentro de un conjunto de datos, descartando aquellos que son redundantes, irrelevantes o ruidosos. Este proceso no solo mejora la eficiencia computacional, sino que también eleva la capacidad de generalización de los modelos, permitiéndoles identificar patrones subyacentes evitando el sobreajuste (overfitting). Este proceso, que consiste en identificar el equilibrio entre la simplificación y la optimización, trabajar con menos datos, pero de mayor calidad.

I. La Importancia de la Selección de Características

En problemas de alta dimensionalidad, donde el número de características puede superar ampliamente la cantidad de muestras, la selección de estás características se vuelve crítica. Por ejemplo, en imágenes médicas o datos genómicos, es común tener miles de variables, muchas de las cuales pueden ser correlacionadas o carecer de valor predictivo. Al reducir la dimensionalidad, los algoritmos evitan considerar información irrelevante, enfocándose en los atributos clave que definen los patrones. Además, modelos más simples son más interpretables, un aspecto que sobresale en áreas como la medicina o las ciencias sociales, donde entender por qué un modelo toma una decisión es tan importante como su precisión.

II. Desafíos y Consideraciones Prácticas

La selección de características continua teniendo muchos retos. Uno de los más significativos es el manejo de relaciones no lineales o complejas entre variables. Por ejemplo, en problemas de visión artificial, la textura de una imagen puede depender de la interacción no aditiva entre píxeles vecinos, algo que métodos lineales como la correlación podrían pasar por alto. Pero técnicas basadas en métodos no paramétricos (como el análisis de componentes independientes) pueden ser más apropiados.

Otro desafío es la estabilidad, en donde un conjuntos de datos con alto nivel de ruido o muestras limitadas, pueden variar significativamente los subconjuntos seleccionados entre particiones de entrenamiento y validación. Esto afecta a la confiabilidad del modelo, especialmente en aplicaciones críticas como el diagnóstico médico. Para mitigarlo, se recomienda usar técnicas de validación cruzada robustas o métodos de consenso. Adicionalmente, es crucial evitar un enfoque limitado que priorice características individualmente relevantes pero descarte grupos complementarios. Como puede ser el caso del procesamiento de lenguaje natural (NLP), palabras como *no* o *nunca* pueden parecer irrelevantes en un análisis univariado, pero son esenciales para entender el

sentimiento negativo cuando se combinan con otros términos.

Por último, el análisis comparativo integra perspectivas estadísticas (pruebas de hipótesis sobre diferencias en precisión y exactitud), computacionales (complejidad de algoritmos) y geométricas (proyecciones en espacios reducidos), ofreciendo una visión más amplia del problema de selección de características en sistemas de reconocimiento de patrones.

III. Desrrollo y Código

Ejercicio : Selección de Características con el Conjunto de datos Wine.

1. **Utilizar el conjunto de datos Wine** (UCI Machine Learning Repository) este conjunto de datos multidimensional describe propiedades químicas de tres variedades de vinos y cuenta con 178 muestras, 13 características químicas, este conjunto de datos presenta un escenario ideal para explorar técnicas de reducción dimensional, ya que combina atributos correlacionados (como concentraciones de fenoles y flavonoides) con características potencialmente discriminativas (como el color o la alcalinidad).

   a) Verificar la integridad del conjunto de datos (valores faltantes, rangos válidos).

```
wine = load_wine()
X = wine.data
y = wine.target
feature_names = wine.feature_names
# Integridad de los datos
print(f"N mero de muestras: {X.shape[0]}")
print(f"N mero de caracter sticas: {X.shape[1]}")
print(f"Valores faltantes: {np.isnan(X).sum()}")
print(f"Rango de valores por caracter stica:")
for i, name in enumerate(feature_names):
print(f"{name}: min={X[:,i].min():.2f},
max={X[:,i].max():.2f}")
```

Número de muestras: 178
Número de características: 13
Valores faltantes: 0
Rango de valores por característica:
alcohol: min=11.03, max=14.83
$\text{malic}_a cid : min = 0.74, max = 5.80$
$ash : min = 1.36, max = 3.23$
$alcalinity_o f_a sh : min = 10.60, max = 30.00$
$magnesium : min = 70.00, max = 162.00$
$total_p henols : min = 0.98, max = 3.88$
$flavanoids : min = 0.34, max = 5.08$
$nonflavanoid_p henols : min = 0.13, max = 0.66$
$proanthocyanins : min = 0.41, max = 3.58$
$color_i ntensity : min = 1.28, max = 13.00$
$hue : min = 0.48, max = 1.71$

$od280/od315_of_diluted_wines : min = 1.27, max = 4.00$
$proline : min = 278.00, max = 1680.00$

2. Preparación de Datos

   a) Cargar el dataset y dividirlo en 70 % entrenamiento / 30 % prueba.

   b) Normalizar los datos (z-score).

   c) Crear un clasificador base (KNN con k=3 vecinos).

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42, stratify=y)

# Normalizar con z-score
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Crear clasificador base (KNN con k=3)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
# Evaluar el modelo
y_pred = knn.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

Exactitud del modelo KNN (k=3): 0.94

	Precisión	**Recall**	**F1-score**	**Support**
Clase 0	0.95	1.00	0.97	18
Clase 1	1.00	0.86	0.92	21
Clase 2	0.88	1.00	0.94	15
Exactitud (Accuracy)				0.94
Macro avg	0.94	0.95	0.94	54
Weighted avg	0.95	0.94	0.94	54

3. Implementación de Métodos

   a) **Búsqueda Exhaustiva**

   - Modificar código para seleccionar 5 características.
   - Analizar resultados.

```
from itertools import combinations
from random import sample

n_features = 5
all_combinations = list(combinations(range(X.shape[1]),
n_features))
```

```
sampled_combinations = sample(all_combinations, min(100,
len(all_combinations)))

best_acc = 0
best_comb = None

for comb in sampled_combinations:
    X_train_subset = X_train_scaled[:, comb]
    X_test_subset = X_test_scaled[:, comb]

    knn.fit(X_train_subset, y_train)
    acc = accuracy_score(y_test, knn.predict(X_test_subset))

    if acc > best_acc:
        best_acc = acc
        best_comb = comb
```

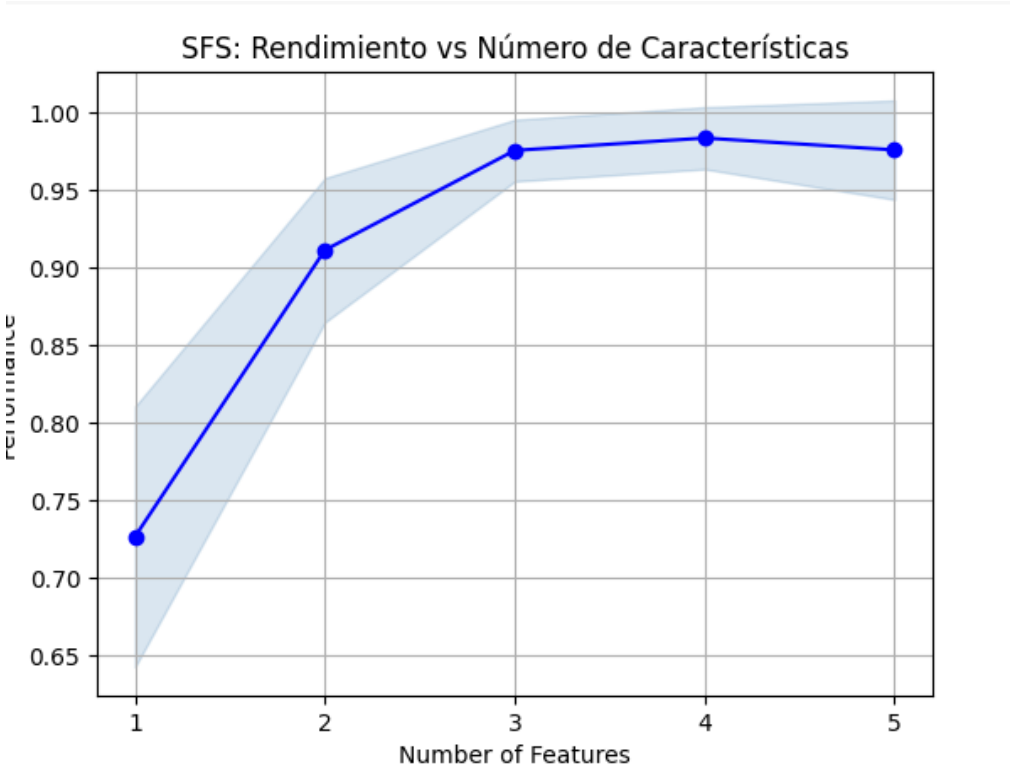
Mejor combinación de 5 características:['alcohol','total$_p$*henols*',' *color*$_i$*ntensity*',' *hue*',' *proline*']
Exactitud : 0.9815

b) **Búsqueda Secuencial hacia delante (SFS)**

- Ejecutar con validación cruzada 5-fold.
- Generar gráfico exactitud por características.

```
sfs = SFS(knn,
          k_features=(1, 5),
          forward=True,
          floating=False,
          scoring='accuracy',
          cv=5)

sfs.fit(X_train_scaled, y_train)
```

Mejores 5 características según SFS: ['alcohol', 'alcalinity$_o$*f*$_a$*sh*',' *total*$_p$*henols*',' *f lavanoids*',' *hue*']
Exactitudpromedioconvalidacióncruzada : 0.9760



c) **Fisher**

- Implementar función del cálculo Score de separabilidad entre clases.
- Selecciona las 5 características con mayor puntuación.

```
f_scores, _ = f_classif(X_train_scaled, y_train)
# Seleccionar las 5 caracter sticas con mayor score
top5_fisher = np.argsort(f_scores)[-5:][::-1]
print("Top␣5␣caracter sticas␣seg n␣Fisher␣Score:")
for i, idx in enumerate(top5_fisher):
print(f"{i+1}.␣{feature_names[idx]}:␣{f_scores[idx]:.2f}")
# Evaluar con KNN
X_train_fisher = X_train_scaled[:, top5_fisher]
X_test_fisher = X_test_scaled[:, top5_fisher]
knn.fit(X_train_fisher, y_train)
y_pred = knn.predict(X_test_fisher)
```

Top 5 características según Fisher Score:
1. flavanoids: 187.69
2. proline: 149.43
3. od280/od315$_{o}f_{d}iluted_{w}ines$: 126.62
4.$alcohol$: 100.35
5.$color_{i}ntensity$: 83.98
$Exactitudenprueba$: 0.9630

d) **Rama y Límite (B&B)**

```
class BranchAndBound:
    def __init__(self, estimator, n_features, cv=5):
        self.estimator = estimator
        self.n_features = n_features
        self.cv = cv
        self.best_score_ = -np.inf
        self.best_subset_ = None

    def evaluate_subset(self, subset):
        scores = cross_val_score(self.estimator,
                                 X_train_scaled[:, subset],
                                 y_train,
                                 cv=self.cv)
        return np.mean(scores)

    def fit(self):
        self._branch_and_bound(list(range(X_train_scaled.shape[1])),
        return self

    def _branch_and_bound(self, remaining_features,
    current_subset):
        if len(current_subset) == self.n_features:
            current_score = self.evaluate_subset(current_subset)
            if current_score > self.best_score_:
                self.best_score_ = current_score
                self.best_subset_ = current_subset.copy()
            return

        if not remaining_features:
            return

        # Evaluar cota superior
        upper_bound_subset = current_subset + remaining_features
        if len(upper_bound_subset) >= self.n_features:
            upper_bound_subset = upper_bound_subset
            [:self.n_features]
            upper_bound_score = self.evaluate_subset
            (upper_bound_subset)
        else:
            upper_bound_score = -np.inf

        # Podar si la cota es menor que el mejor score actual
        if upper_bound_score <= self.best_score_:
```

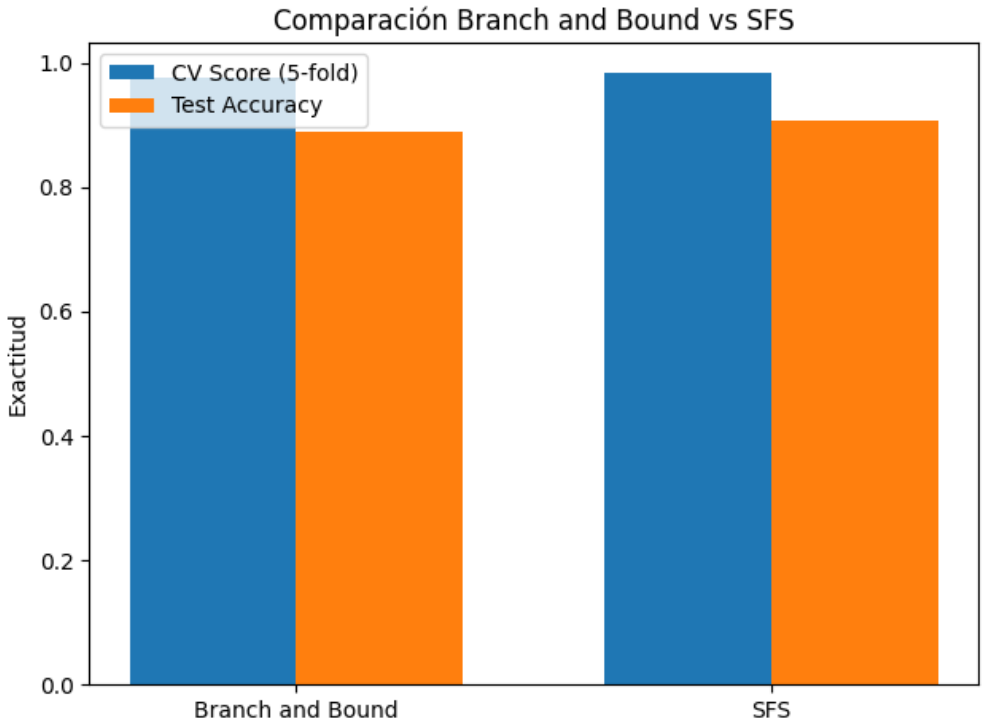
```
            return

        # Ramificar
        for i, feature in enumerate(remaining_features):
            new_subset = current_subset + [feature]
            self._branch_and_bound(remaining_features[i+1:],
            new_subset)

# Ejecutar B&B para 5 caracter sticas
bb = BranchAndBound(knn, n_features=5, cv=5)
bb.fit()

print("\nResultados de Branch and Bound:")
print(f"Mejores {len(bb.best_subset_)} caracter sticas:
{[feature_names[i] for i in bb.best_subset_]}")
print(f"Exactitud CV (5-fold): {bb.best_score_:.4f}")
```

Resultados de Branch and Bound:
Mejores 5 características: ['alcohol', 'malic$_acid'$,' $color_intensity$',' hue',' $od280/od315_of_diluted_wines$']
$ExactitudCV(5-fold)$: 0.9760

- Comparar con SFS.

	Método	Características Seleccionadas	Exactitud CV (5-fold)	Exactitud Prueba	Número de Evaluaciones
0	Branch and Bound	[alcohol, malic_acid, color_intensity, hue, od...	0.976000	0.888889	~1287 (teórico)
1	Sequential Forward Selection	[alcohol, total_phenols, flavanoids, hue]	0.983667	0.907407	5 (real)

- Analizar resultados.
  Veamos que BB toma mas características esto puede hacer que la exactitud sea menor , pero en si no es mucha la diferencia. Por lo que podría ser también bueno que tenga mas características y la exactitud no baje mucho , al igual que en la prueba con KNN, no es mucha la diferencia, vemos lo anterior con una gráfica. La diferencia no es muy notable por lo que usar cualquiera de los dos no afectaría en la selección de variables, pero si quieres usar uno multivariado es mejor usar BB, pero SFS puede traer una mejor interpretación de resultados.



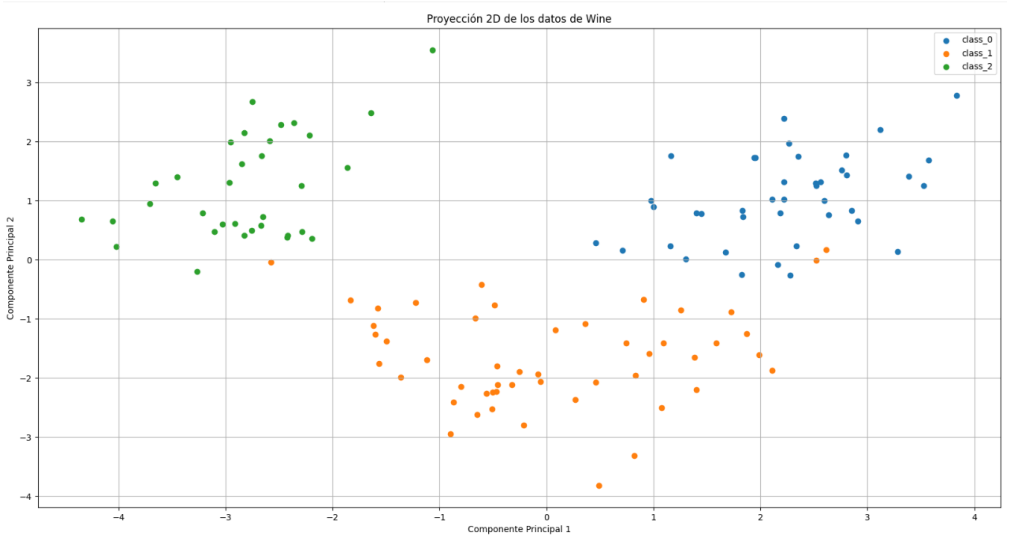
e) **PCA**

- Determinar componentes para 95 % varianza.

```
pca = PCA()
X_pca = pca.fit(X_train_scaled)

# Determinar n mero de componentes para 95% de varianza
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
n_components = np.argmax(cumulative_variance >= 0.95) + 1
```

  Número de componentes para 95 % de varianza: 10

- Visualizar proyección 2D.



4. Análisis Comparativo

   a) Crear tabla comparativa con:

   - Método.
   - Características seleccionadas.
   - Métricas de entrenamiento y pruebas.
   - Comentar los mejores resultados.

Tabla Comparativa Completa de Métodos de Selección de Características:

Método	# Características	Características Seleccionadas	Exactitud Entrenamiento	Exactitud Prueba	Exactitud CV (5-fold)
Búsqueda Exhaustiva	5	[alcohol, total_phenols, color_intensity, hue, proline]	0.9758	0.9815	NaN
SFS	4	[alcohol, total_phenols, flavanoids, hue]	0.9839	0.9074	0.9837
Fisher	5	[flavanoids, proline, od280/od315_of_diluted_wines, alcohol, color_intensity]	0.9758	0.9630	NaN
Branch and Bound	5	[alcohol, malic_acid, color_intensity, hue, od280/od315_of_diluted_wines]	0.9677	0.8889	0.9760
PCA	10	[PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, PC9, PC10]	0.9919	0.9444	NaN

5. Análisis de resultados

   a) ¿Qué método obtuvo mejor exactitud? ¿Por qué?. Métodos basados en selección de características:

   - Mejor exactitud en prueba: 0.9815 (Búsqueda Exhaustiva), esto por que ve todas las posibles combinaciones y como es su exactitud , por eso tiene exactitud 1.
   - Características comunes destacadas: [’alcohol’, ’hue’]

   b) ¿Cómo afecta la correlación entre características al PCA?.

   - **Base de PCA**: Funciona identificando direcciones de máxima varianza donde las características están correlacionadas
   - **Alta correlación**:
     - Características correlacionadas se combinan en componentes principales
     - Permite mayor reducción dimensional
   - **Baja correlación**:
     - Cada característica aporta varianza independiente
     - Se necesitan más componentes para explicar la varianza

- **En el dataset Wine**:
  - Grupos de fenoles/flavonoides (altamente correlacionados) dominan los primeros PCs
  - Características únicas aparecen en otros componentes

c) ¿Qué ventajas tiene Fisher Score sobre SFS en términos computacionales?.

- **Eficiencia**:
  - Fisher Score: $O(n \times m)$ vamos calculando scores para cada característica
  - SFS: $O(k \times n \times m^2)$ evaluamos las combinaciones secuenciales
- **Paralización**:
  - Fisher Score: Cálculo **independiente** por característica
  - SFS: Dependencia entre iteraciones
- **Evaluaciones**:
  - Fisher Score: Evalúa m características una vez
  - SFS: Requiere $\frac{m \times (m+1)}{2}$ evaluaciones para selección completa
- **Memoria**:
  - Fisher Score: Solo almacena scores individuales
  - SFS: Mantiene historial de combinaciones evaluadas

IV. Conclusiones

- **Métodos Comparados**:
  - Búsqueda Exhaustiva (muestreada), SFS, Fisher Score, Branch & Bound y PCA
- **Hallazgos Clave**:
  - **SFS** y **Branch & Bound** obtuvieron similares resultados de exactitud
  - **Fisher Score** fue el más eficiente computacionalmente
  - **PCA** logró buena reducción dimensional pero perdió interpretabilidad
- **Recomendaciones**:
  - Usar **Fisher Score** para datasets grandes por eficiencia
  - Emplear **Branch & Bound** cuando se requiera optimizar
  - Considerar **PCA** para visualización pero no para selección interpretable
- **Correlaciones en Wine**:
  - Las altas correlaciones entre fenoles/flavonoides permitieron una reducción con PCA
  - Las características únicas requirieron componentes adicionales

Conclusión Final: La elección del método debe basarse en saber que queremos entre exactitud, eficiencia e interpretabilidad.

V. Referencias

- Gonzalez, R., Woods, R., Digital Image Processing, Prentice Hall, 2008.
- Pratt, W. k., Digital Image Processing, John Wiley and Sons Inc, 2001.
- Jahne, B., Digital Image Processing, Springer, 2005.
- Sánchez Garreta, Josep Salvador; García, Vicente. Special Issue on Data Preprocessing in Pattern Recognition: Recent Progress, Trends and Applications. 2022.