IRIS con NaiveBayes y Cross Validation

Adrián Yared Armas de la Nuez

Contenido

1. Enunciado

El objetivo de esta actividad es observar las diferencias en la Predicción en los modelos de NaiveBayes al utilizar o no la Validación cruzada en el entrenamiento de los modelos. Para ello anotaremos los resultados en la siguiente tabla comparativa:

	Predicción	
Modelo	Sin CrossValidation	Con CrossValidation
GaussianNB	XX	yy
MultiNomialNB	XX	YY
BernouilliNB	XX	YY
ComplementNB	XX	YY
CategoricalNB	XX	YY

En la primera columna indicamos los resultados obtenidos en la actividad 3.2 – Comparativa clasificadores NaiveBayes Como referencia se pueden utilizar los siguientes ejemplos:

Título: Ejemplo_3_2_Iris_NaiveBayes - GaussianNB.ipynb Url: https://colab.research.google.com/drive/1tjF8jjv33sGhgTL3UjWlQh0iwMypjhAj?usp=sharing

Título: Ejemplo_3_3_Clasificación_con_Naive_Bayes_(Heart_Diseases).ipynb Url:https://colab.research.google.com/drive/1hwri6X-N_cHmpZs31-zyK2XwRyfA4EGN?usp=sharing

Una vez realizados los ajustes y el entrenamiento, concluir si hay diferencias entre utilizar o no la Validación Cruzada (Cross Validation).

2. Actividad

2.1 Código explicado

2.1.1 Imports

```
# Imports
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB,
ComplementNB, CategoricalNB
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.metrics import accuracy_score
```

2.1.2 Carga de los datos

```
# Iris data Load
data = load_iris()
X, y = data.data, data.target
```

2.1.3 División en train y test

```
# Train and test data split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

2.1.4 Modelos Naive Bayes

```
# Naive Bayes Models
models = {
    "GaussianNB": GaussianNB(),
    "MultinomialNB": MultinomialNB(),
    "BernoulliNB": BernoulliNB(),
    "ComplementNB": ComplementNB(),
}
```

2.1.5 Discretización de los datos para CategoricalNB

```
# Para CategoricalNB, data discretization
categorical_encoder = KBinsDiscretizer(n_bins=10, encode='ordinal',
strategy='uniform')
X_categorical = categorical_encoder.fit_transform(X)
models["CategoricalNB"] = CategoricalNB()
```

2.1.7 Tabla solicitada

Estructura:

```
# Results
results = {"Model": [], "Without Cross-Validation": [], "With
Cross-Validation": []}
```

Relleno de los resultados:

```
for name, model in models.items():
    # with NO cross validation
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    # with cross validation
    if name == "CategoricalNB":
        cv_scores = cross_val_score(model, X_categorical, y, cv=5)
    else:
        cv_scores = cross_val_score(model, X, y, cv=5)
```

Guardado en la tabla:

```
    # Save results
    results["Model"].append(name)
    results["Without Cross-Validation"].append(accuracy)
    results["With Cross-Validation"].append(cv_scores.mean())
```

2.1.8 Mostrar los resultados

```
# Show results
import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)
```

2.2 Código completo

```
# Imports
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB,
ComplementNB, CategoricalNB
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.metrics import accuracy_score

# Iris data Load
data = load_iris()
X, y = data.data, data.target

# Train and test data split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Naive Bayes Models
models = {
    "GaussianNB": GaussianNB(),
    "MultinomialNB": MultinomialNB(),
    "BernoulliNB": BernoulliNB(),
    "ComplementNB": ComplementNB(),
}

# Para CategoricalNB, data discretization
categorical_encoder = KBinsDiscretizer(n_bins=10, encode='ordinal',
strategy='uniform')
X_categorical = categorical_encoder.fit_transform(X)
models["CategoricalNB"] = CategoricalNB()

# Results
results = {"Model": [], "Without Cross-Validation": [], "With
Cross-Validation": []}

for name, model in models.items():
    # with NO cross validation
```

```
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    # with cross validation
    if name == "CategoricalNB":
        cv_scores = cross_val_score(model, X_categorical, y, cv=5)
    else:
        cv_scores = cross_val_score(model, X, y, cv=5)

    # Save results
    results["Model"].append(name)
    results["Without Cross-Validation"].append(accuracy)
    results["With Cross-Validation"].append(cv_scores.mean())

# Show results
import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)
```

2.3 Justificación de los resultados

Los resultados obtenidos parecen razonables y están alineados con el comportamiento esperado de los diferentes modelos Naive Bayes al aplicarse al conjunto de datos Iris.

GaussianNB
Trabaja bien con datos continuos y distribuciones gaussianas.
Resultados esperados: Una precisión alta (generalmente >90%), tanto con como sin validación cruzada.
Los resultados para GaussianNB (1.0 sin CV, 0.9533 con CV) son consistentes y razonables.

MultinomialNB
Este modelo es más adecuado para datos discretos o de conteo
Una precisión algo menor que GaussianNB
Los resultados (0.9 sin CV, 0.9533 con CV) sugieren que MultinomialNB se beneficia de la validación cruzada. Esto es plausible, ya que la CV suaviza posibles sesgos de la partición inicial.

BernoulliNB
Este modelo es más adecuado para datos binarios
Precisión significativamente baja
Los resultados (0.3 sin CV, 0.3333 con CV) reflejan este comportamiento esperado

ComplementNB
Es una variante de MultinomialNB diseñada para clases desbalanceadas
Un desempeño intermedio, mejor que BernoulliNB pero posiblemente inferior a GaussianNB
Los resultados (0.7 sin CV, 0.6667 con CV) son razonables para este caso

CategoricalNB
Este modelo está diseñado específicamente para datos categóricos. Para usarlo con Iris, discretizar los datos es necesario (como hice en el código con KBinsDiscretizer)
Una precisión muy alta, pero algo menor que GaussianNB, ya que los datos de Iris no son naturalmente categóricos, que es la especialidad de este
Los resultados (0.9667 sin CV, 0.94 con CV) son consistentes y razonables si los datos fueron discretizados correctamente

3. Github y Colab