diplomatura universitaria en
inteligencia artificial

FACULTAD DE CIENCIAS EXACTAS
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Introducción a Python

Librería SciKit Learn – Modelos de clasificación

Agenda

- Introducción
- Entrenar un modelo
- Predicciones y evaluación
- Pipelines
- Métricas
- Lectura de datasets

Scikit-learn

- Librerías de herramientas para minería, análisis de datos y Machine Learning
  - Preprocesamiento
  - Clasificación
  - Regresión
  - Clustering
  - Reducción de dimensionalidad

Scikit-learn

- Clasificación
  - Árboles de decisión
    - Sklearn.tree.DecisionTreeClassifier
  - Support Vector Machines
    - sklearn.svm.SVC
  - Nearest Neighbors
    - sklearn.neighbors.KNeighborsClassifier
    - sklearn.neighbors.NearestCentroid
  - Naive Bayes
    - sklearn.naive_bayes.GaussianNB
    - sklearn.naive_bayes.MultinomialNB
    - sklearn.naive_bayes.CategoricalNB

Entrenar un modelo

```
>>> from sklearn.tree import DecisionTreeClassifier
>>> clf = DecisionTreeClassifier()
>>> X = [[ 1, 2, 3], # 2 muestras (filas), 3 atributos (columnas
...      [11, 12, 13]]
>>> y = [0, 1] # clases de cada elemento

>>> clf.fit(X, y)
DecisionTreeClassifier(…)
```

Hacer predicciones

```
# Predice las clases sobre el mismo set de entrenamiento
>>> clf.predict(X)
array([0, 1])

# Predice las clases sobre nuevos datos
>>> clf.predict([[4, 5, 6], [14, 15, 16]])
array([0, 1])
```

Pipelines

- Transformers y predictores pueden ser combinados en un objeto unificado: un Pipeline.
    - Provee la misma interface que un predictor: fit y predict.

```
>>> from sklearn.preprocessing import MinMaxScaler
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.metrics import accuracy_score
>>> # Creamos el pipeline
>>> pipe = make_pipeline(
...             MinMaxScaler()
...             RandomForestClassifier(random_state=0)
)
```

Pipelines

```
>>> X = [[ 1, 2, 3], # 2 ejemplos (filas), 3 atributos (columnas
 ...      [11, 12, 13]]
>>> y = [0, 1] # clases de cada elemento
...
>>> # llama al metodo fit_transform del pipeline
>>> pipe.fit_transform(X, y)

>>> # utiliza el pipeline para predecir nuevos ejemplos
>>> pipe.predict([[4, 5, 6], [14, 15, 16]])
```

Training y testing

- Split en training y testing

```
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.datasets import load_iris
>>> X_train, X_test, y_train, y_test = train_test_split(iris.data,
                                                        iris.target, test_size=0.4, random_state=0)
>>> clf = tree.DecisionTreeClassifier()
>>> clf.fit(X_train, y_train)
>>> clf.score(X_test, y_test)
0.95
```

split

training

testing

Cross validation

- Cross validation

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.tree import DecisionTreeClassifier
>>> clf = DecisionTreeClassifier(random_state=0)
>>> iris = load_iris()
>>> cross_val_score(clf, iris.data, iris.target, cv=10)
array([1.0, 0.93333333, 1.0, 0.93333333, 0.93333333,
       0.86666667, 0.93333333, 1.0, 1.0, 1.0])
```

Cross validation

- Cross validation

```
>>> clf = DecisionTreeClassifier(random_state=0)
>>> iris = load_iris()
>>> score = cross_val_score(clf, iris.data, iris.target,
                            cv=10)
>>> print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
                            scores.std()))
Accuracy: 0.96 (+/- 0.09)
```

Cross validation con múltiples métricas

```
>>> from sklearn.model_selection import cross_validate
...
>>> scoring = ['precision_macro', 'recall_macro']
>>> scores = cross_validate(clf, iris.data, iris.target,
                            scoring=scoring, cv=5)
>>> sorted(scores.keys())
['fit_time', 'score_time', 'test_precision_macro',
'test_recall_macro']
>>> scores['test_recall_macro']
array([0.96..., 1. ..., 0.96..., 0.96..., 1. ])
```

Cross validation con múltiples métricas

- Listar las métricas que se pueden computar con `cross_validate`

```
>>> from sklearn.metrics import SCORERS
>>> sorted(SCORERS.keys());
['accuracy', ...
 'f1_macro', ...
 'precision',
 'precision_macro',
 'precision_weighted',
 'recall_macro', ...
 'roc_auc', ...]
```

Predicciones por cross validation

- Predicciones
  - `clf.predict(X_test)`
    - Modelo entrenado (fit)
  - Predicción por cross validation

```
>>> from sklearn
>>> y_pred = cro
        iris.ta
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0,..., 2, 1, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2])
```

Reporte en base a las predicciones

- Reporte por clases en base a las predicciones generadas

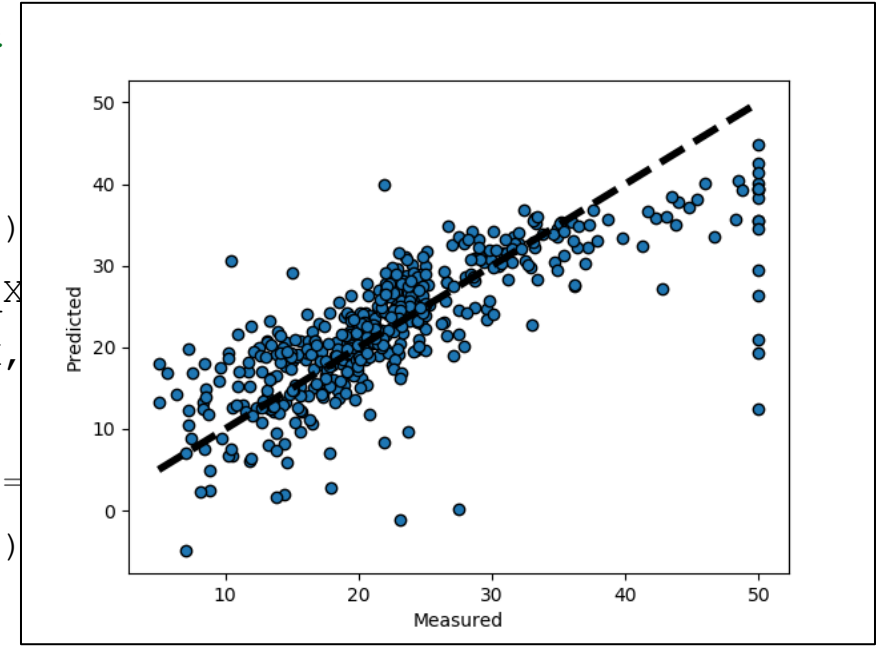
```
>>> from sklearn.metrics import classification_report

>>> print(classification_report(iris.target, y_pred,
        target_names=iris.target_names))
```

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	50
versicolor	0.94	0.94	0.94	50
virginica	0.94	0.94	0.94	50
accuracy			0.96	150
macro avg	0.96	0.96	0.96	150
weighted avg	0.96	0.96	0.96	150

Gráfico de predicciones

```
>>> from sklearn import datasets
>>> from sklearn.model_selection import
>>> from sklearn import linear_model
>>> import matplotlib.pyplot as plt
>>> lr = linear_model.LinearRegression()
>>> X, y = datasets.load_boston(return_X
>>> predicted = cross_val_predict(lr, X,
>>> fig, ax = plt.subplots()
>>> ax.scatter(y, predicted, edgecolors=
>>> ax.plot([y.min(), y.max()], [y.min()
>>> ax.set_xlabel('Measured')
>>> ax.set_ylabel('Predicted') plt.show()
```



Matriz de confusión

Split

```
>>> from sklearn.metrics import confusion_matrix
...
>>> y_pred = classifier.fit(X_train,y_train).predict(X_test)
>>> conf_mat = confusion_matrix(y_test, y_pred)
```

Cross validation

```
>>> y_pred = cross_val_predict(classifier, iris.data,
iris.target, cv=5)
>>> conf_mat = confusion_matrix(iris.targe
```

```
[[50  0  0]
 [ 0 48  2]
 [ 0 10 40]]
```

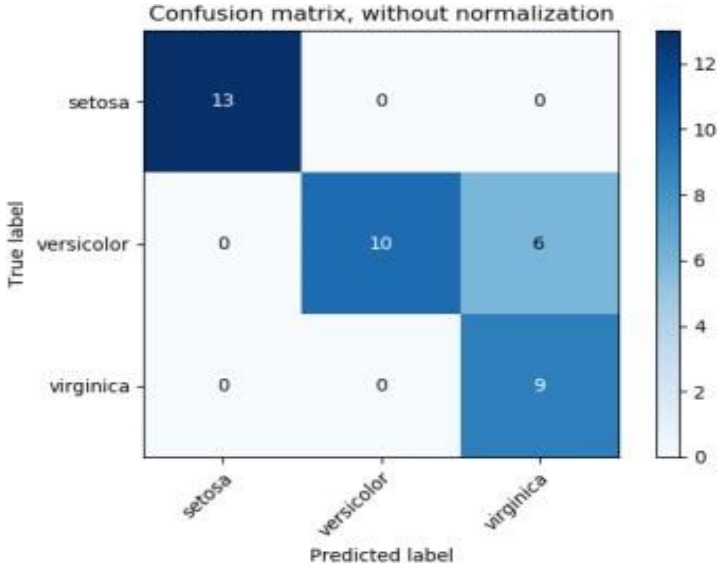
Matriz de confusión

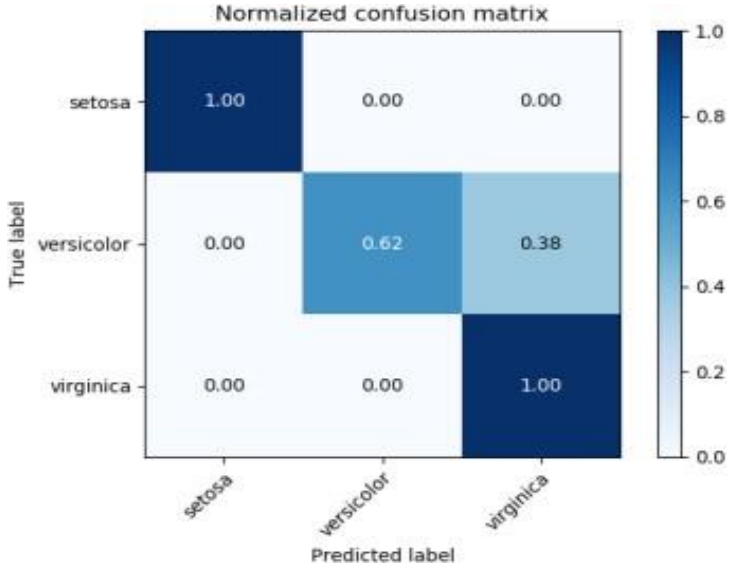
Gráfico usando `plot_confusion_matrix`

```
>>> from sklearn.metrics import plot_confusion_matrix
>>> class_names = iris.target_names
...
>>> disp = plot_confusion_matrix(classifier, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
>>> disp.ax_.set_title(title)
>>> print(title)
>>> print(disp.confusion_matrix)
```

Matriz de confusión

Gráfico usando `plot_confusion_matrix`





Carga de datasets

- Sk-learn provee datasets propios
  - `sklearn.datasets`
    - Iris (Clasificación)
    - Boston (Regresión)
    - Diabetes(Regresión)
    - Breast_Cancer (Clasificación)
- OpenML.org
- Google Drive

Datasets en openml

- Datasets openml https://www.openml.org

```
>>> from sklearn.datasets import fetch_openml
>>> from sklearn.tree import DecisionTreeClassifier
>>> from sklearn.model_selection import cross_val_score

>>> plants = fetch_openml(name='one-hundred-plants-shape')
>>> clf = DecisionTreeClassifier()
>>> scores = cross_val_score(clf, plants.data,
   plants.target, cv=10)
>>> scores.mean()
```

Datasets en Google Drive

- Dataset en drive

```
>>> from google.colab import drive
>>> drive.mount('/content/gdrive')
```

Requiere permisos que son solicitados la primera vez que se ejecuta en el entorno de ejecución

```
>>> import csv
>>> with open("/content/gdrive/My Drive/Colab
        Notebooks/dataMLxtend.txt", 'r') as f:
>>>   reader = csv.reader(f)
>>>   dataset = list(reader)
```

Datasets en Google Drive

- Lectura de un conjunto de jsons

```
>>> from google.colab import drive
>>> drive.mount('/content/gdrive')

>>> import glob
>>> import json
>>> file_path = glob.glob("/content/gdrive/My
        Drive/Colab Notebooks/jsons/*.json")
>>> for file in file_path:
>>>    with open(file, 'r') as j:
>>>      json_data = json.load(j)
>>>      print(json_data)
```

Enlaces útiles

- https://scikit-learn.org/stable/supervised_learning.html
- https://scikit-learn.org/stable/modules/cross_validation.html
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html