How do you perform Quadratic Discriminant Analysis in Python step-by-step?

Authored by

stats writer

April 21, 2024

Quadratic Discriminant Analysis (QDA) is a statistical method used for classification tasks, which involves identifying the underlying relationship between a set of independent variables and a categorical target variable. This method is particularly useful for datasets with non-linear relationships between the variables and the target.

In order to perform QDA in Python, the following steps can be followed:

1. Import the necessary libraries: First, import the necessary libraries such as numpy, pandas, and scikit-learn to perform QDA.

2. Load the dataset: Load the dataset into your Python environment using pandas. Make sure the target variable is categorical.

3. Split the dataset: Split the dataset into training and testing sets using the train_test_split function from scikit-learn. This will help in evaluating the performance of the model.

4. Standardize the data: Since QDA involves calculating the mean and variance of the independent variables, it is important to standardize the data using the StandardScaler function from scikit-learn.

5. Fit the model: Use the QuadraticDiscriminantAnalysis function from scikit-learn to fit the model on the training data.

6. Make predictions: Use the predict function to make predictions on the testing data.

7. Evaluate the model: Calculate the accuracy of the model using the accuracy_score function from scikit-learn.

8. Fine-tune the model: If the model performance is not satisfactory, try fine-tuning the model by adjusting the regularization parameters.

9. Make final predictions: Once the model is fine-tuned, make final predictions on the test data.

10. Visualize the results: Finally, plot the results to understand the performance of the model visually.

By following these steps, you can perform Quadratic Discriminant Analysis in Python and use it for classification tasks with non-linear relationships between the variables and the target.

Quadratic Discriminant Analysis in Python (Step-by-

Step)

Quadratic discriminant analysis is a method you can use when you have a set of predictor variables and you'd like to classify a **response variable** into two or more classes.

It is considered to be the non-linear equivalent to **linear discriminant analysis**.

This tutorial provides a step-by-step example of how to perform quadratic discriminant analysis in Python.

Step 1: Load Necessary Libraries

First, we'll load the necessary functions and libraries for this example:

```
from sklearn.model_selectionimport train_test_split
from sklearn.model_selectionimport RepeatedStratifiedKFold
from sklearn.model_selectionimport cross_val_score
from sklearn.discriminant_analysisimport QuadraticDiscriminantAnalysis
from sklearn import datasets
import matplotlib.pyplotas plt
```

```
import pandas as pd
import numpy as np
```

Step 2: Load the Data

For this example, we'll use the iris dataset from the sklearn library. The following code shows how to load this dataset and convert it to a pandas DataFrame to make it easy to work with:

```
#load iris dataset
iris = datasets.load_iris()

#convert dataset to pandas DataFrame
df = pd.DataFrame(data = np.c_, iris],
columns = iris + )
df = pd.Categorical.from_codes(iris.target, iris.target_names)
df.columns =

#view first six rows of DataFrame
df.head()
```

```
s_length s_width p_length p_width target species
0 5.1 3.5 1.4 0.2 0.0 setosa
1 4.9 3.0 1.4 0.2 0.0 setosa
```

2 4.7 3.2 1.3 0.2 0.0 setosa

3 4.6 3.1 1.5 0.2 0.0 setosa

4 5.0 3.6 1.4 0.2 0.0 setosa

#find how many total observations are in datasetlen(df.index)

150

We can see that the dataset contains 150 total observations.

For this example we'll build a quadratic discriminant analysis model to classify which species a given flower belongs to.

We'll use the following predictor variables in the model:

Sepal lengthSepal widthPetal lengthPetal width

And we'll use them to predict the response variable *Species*, which takes on the following three potential classes:

setosaversicolorvirginica

Step 3: Fit the QDA Model

Next, we'll fit the QDA model to our data using the QuadraticDiscriminantAnalsyis function from sklearn:

```
#define predictor and response variables
X = df]
y = df

#Fit the QDA model
model = QuadraticDiscriminantAnalysis()
model.fit(X, y)
```

Step 4: Use the Model to Make Predictions

Once we've fit the model using our data, we can evaluate how well the model performed by using repeated stratified k-fold cross validation.

For this example, we'll use 10 folds and 3 repeats:

```
#Define method to evaluate model
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
random_state=1)

#evaluate model
scores = cross_val_score(model, X, y,
```

scoring='accuracy', cv=cv, n_jobs=-1)
print(np.mean(scores)) 0.97333333333334

We can see that the model performed a mean accuracy of 97.33%.

We can also use the model to predict which class a new flower belongs to, based on input values:

#define new observation
new =

#predict which class the new observation belongs to
model.predict()

array(, dtype='<U10')

We can see that the model predicts this new observation to belong to the species called *setosa*.

You can find the complete Python code used in this tutorial here.