01418362 Introduction to Machine Learning

BREAST CANCER
NAIVE BAYES CLASSIFIER

6410406738 เนติชัย วงษ์คำภู

Disclaimer

โปรเจคนี้อาจไม่สามารถใช้งานกับปัญหาจริงบนโลก เนื่องจากการวินิจฉัยการเป็นมะเร็งเต้านม อาจมีปัจจัยอื่นๆ ที่ต้องพิจรณาร่วมด้วย ฉะนั้นโปรเจคชิ้นนี้ใช้เพื่อศึกษาการเรียนรู้อัลกอริทึมการเรียนรู้ของเครื่องเท่านั้น

Dataset

Breast Cancer Dataset

Binary Classification Prediction for type of Breast Cancer

kaggle.com

30 features, 1 label

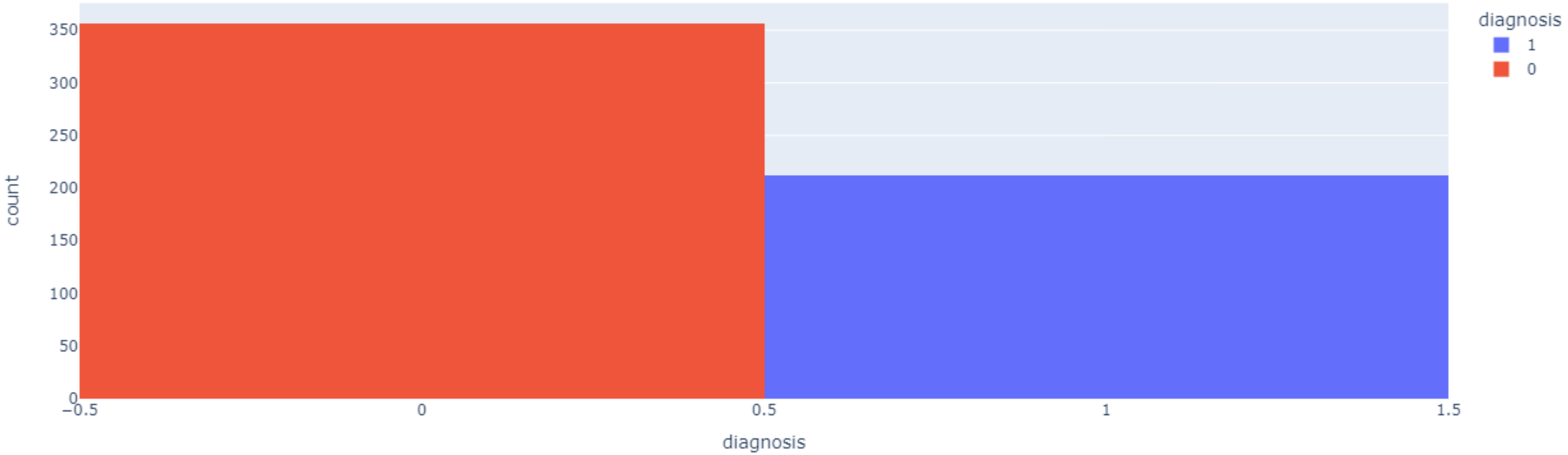
569 records

The Ideas

ใช้ Naive Bayes Classifier ในการเรียนรู้และจำแนกการเป็นมะเร็งเต้านม

Classes

1	Malignant (เนื้อร่าย)	212
0	Benign(เนื้อดี)	357



Data Preprocessing

Label

```
# label into 1/0
df['diagnosis'] = (df['diagnosis'] == 'M').astype(int)
```

Features

```
# collect only features
names = [index for index, value in df.iteritems()]
names.remove('diagnosis')
```

Spliting Data

```
def train_test_split(x,y,test_size):
    i = int((1 - test_size) * x.shape[0])
    #randomly seperate using permutation
    split = np.random.permutation(x.shape[0])

    x_train, x_test = np.split(np.take(x,split,axis=0), [i])
    y_train, y_test = np.split(np.take(y,split), [i])
    return x_train, x_test ,y_train, y_test
```

Standardization

Z-score

$$z = \frac{value - mean}{standard\ deviation}$$

```
def scale(X):
    mean = np.mean(X, axis=0)
    std = np.std(X, axis=0)

    Z = (X - mean)/std
    return Z
```

Learning Algorithm
Gaussian Naive Bayes

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

$$P_\Theta(y|X) \propto \sum_{\alpha=1}^{d} X_\alpha \times \log([\theta_y]_d) + \log(P_\theta(y))$$

```python
def get_probability(self, x):

        # Create an empty list to store the posteriors
        posteriors = []

        for i, c in enumerate(self.unique_classes):
            # Calculate the log of the prior
            prior = np.log(self.priors[i])
            # Calculate the new posterior and append it to the list
            posterior = np.sum(np.log(self.gaussian_density(x, i)))
            posterior = posterior + prior
            posteriors.append(posterior)

        # Return the class with the highest class probability
        return self.unique_classes[np.argmax(posteriors)]
```

Likelihood P(X|y): Gaussian Distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

```python
# prior assumption
    def gaussian_density(self, x, c):

        # Get the mean and the variance for the specified class
        mean = self.mean[c]
        variance = self.variance[c]

        # Calculate the Gaussian density function
        const = 1 / np.sqrt(variance * 2 * np.pi)
        proba = np.exp(-0.5 * ((x - mean) ** 2 / variance))

        return const * proba
```

Prior P(y): Estimate from data

$$P(Y = y) \approx \frac{\sum_{i=1}^{n} I(Y_i = y)}{n}$$

```python
self.priors[i] = X_c.shape[0] / self.m
```

Result

```
model = NaiveBayes()
model.fit(X_train, y_train)
prediction = model.predict(X_test)
accuracy(y_test, prediction)
```

✓ 0.0s

```
0.9298245614035088
```

Accuracy ~ 92.98%

THANK YOU