paper about naive bayes and Gaussian model, Categorical model

Naive bayes :

Naive Bayes is a popular and simple probabilistic classifier based on Bayes' theorem with the "naive" assumption of independence between features. Despite its simplicity, Naive Bayes often performs surprisingly well in many real-world scenarios and is widely used in various applications, especially in text classification and spam filtering. Bayes' theorem, in its essence, encapsulates the probabilistic relationship between two events, typically denoted as A and B. It mathematically articulates how our initial beliefs, represented by the prior probability $P(A)$, are modified in light of new evidence, denoted as $P(B|A)$ and $P(B)$, yielding the posterior probability $P(A|B)$. In other words, it allows us to compute the probability of an event given some observed data, thus enabling informed decision-making in uncertain environments.[1]

The Naive Bayes classifier, an extension of Bayes' theorem, harnesses its probabilistic machinery to tackle classification tasks. Central to its operation is the "naive" assumption of feature independence, which posits that the presence or absence of a particular feature is unrelated to the presence or absence of other features, given the class label. While this assumption may seem overly simplistic or unrealistic in many contexts, its pragmatic utility has been demonstrated across diverse domains, owing to its computational tractability and often satisfactory performance.[1],[2]

Bayes' Theorem:

Bayes' theorem is a fundamental concept in probability theory, used to calculate the probability of a hypothesis given some observed evidence. Mathematically, it can be expressed as [2]:

$$P(A|B)=P(B|A)\times P(A)P(B)P(A|B)=P(B)P(B|A)\times P(A)$$

where:

- $P(A|B)$ is the probability of event A given event B,
- $P(B|A)$ is the probability of event B given event A,
- $P(A)$ and $P(B)$ are the probabilities of events A and B respectively.

Types of Naive Bayes Classifiers:

1. **Gaussian Naive Bayes:** Assumes that continuous features follow a Gaussian distribution.[3]
2. **Multinomial Naive Bayes:** Suitable for discrete features (e.g., word counts for text classification).[4]
3. **Bernoulli Naive Bayes:** Appropriate when features are binary (e.g., presence or absence of a word in text).[5]

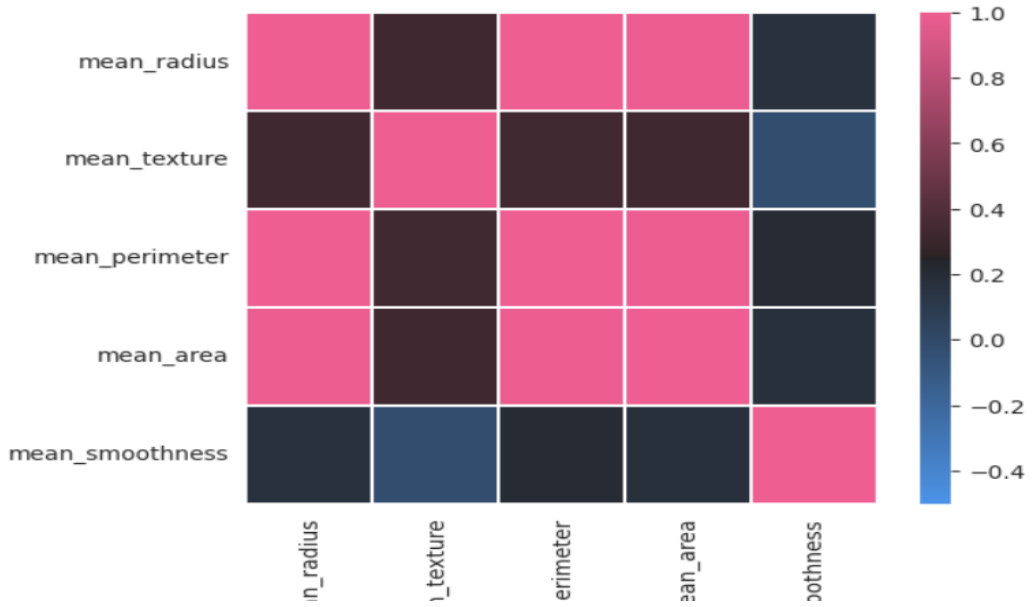
Applications of Naive Bayes:

4. **Text Classification:** Naive Bayes classifiers are widely used for categorizing text documents into predefined categories such as spam detection, sentiment analysis, and topic classification.[3]
5. **Spam Filtering:** Due to its effectiveness in text classification tasks, Naive Bayes is commonly used in email spam filters to classify incoming emails as either spam or non-spam.[3]
6. **Medical Diagnosis:** Naive Bayes classifiers can assist in medical diagnosis by analyzing patient symptoms and predicting possible diseases.[6]
7. **Recommendation Systems:** Naive Bayes classifiers can be used in recommendation systems to predict user preferences based on their past behavior and interactions.[6]
8. **Customer Segmentation:** In marketing, Naive Bayes classifiers can help segment customers into different groups based on their demographics, behavior, or preferences.[6]

Important methods used the code:

```
corr = data.iloc[:,:-1].corr(method="pearson")
cmap = sns.diverging_palette(250,354,80,60,center='dark',as_cmap=True)
sns.heatmap(corr, vmax=1, vmin=-.5, cmap=cmap, square=True, linewidths=.2)
```

```
<Axes: >
```



This code snippet performs the following tasks using Python and popular libraries like Pandas and Seaborn:

1.corr = data.iloc[:,:-1].corr(method="pearson"): This line calculates the Pearson correlation coefficients between all pairs of columns (features) in the
2.cmap = sns.diverging_palette(250,354,80,60,center='dark',as_cmap=True): This line defines a color map using Seaborn's **diverging_palette** function.
3.sns.heatmap(corr, vmax=1, vmin=-.5, cmap=cmap, square=True, linewidths=.2): This line generates a heatmap using Seaborn's **heatmap** function. It visualizes the correlation matrix (**corr**) as a grid of colored squares, where each square's color represents the strength and direction of the correlation between two features.

Calculate P(Y=y) for all possible y

```
[7]  def calculate_prior(df, Y):
         classes = sorted(list(df[Y].unique()))
         prior = []
         for i in classes:
             prior.append(len(df[df[Y]==i])/len(df))
         return prior
```

This function calculates the prior probabilities for each class in a categorical target variable within a DataFrame. It counts the occurrences of each class and divides it by the total number of rows in the DataFrame to get the probability. Then, it returns a list of these probabilities.

The models :

1)Gaussian model: The Gaussian model, also known as the normal distribution, describes data with a bell-shaped curve around its mean. It's characterized by its symmetric shape and is defined by two parameters: the mean and standard deviation. Widely used in statistics and various fields, it represents many natural phenomena and forms the basis for statistical inference, financial modeling, and signal processing due to its simplicity and versatility.[7]

Calculate P(X=x1|Y=y)P(X=x2|Y=y)...P(X=xn|Y=y) * P(Y=y) for all y and find the maximum

```
def naive_bayes_gaussian(df, X, Y):
    # get feature names
    features = list(df.columns)[:-1]

    # calculate prior
    prior = calculate_prior(df, Y)

    Y_pred = []
    # loop over every data sample
    for x in X:
        # calculate likelihood
        labels = sorted(list(df[Y].unique()))
        likelihood = [1]*len(labels)
        for j in range(len(labels)):
            for i in range(len(features)):
                likelihood[j] *= calculate_likelihood_gaussian(df, features[i], x[i], Y, labels[j])

        # calculate posterior probability (numerator only)
        post_prob = [1]*len(labels)
        for j in range(len(labels)):
            post_prob[j] = likelihood[j] * prior[j]

        Y_pred.append(np.argmax(post_prob))

    return np.array(Y_pred)
```

This Python function implements a Gaussian Naive Bayes classifier. Here's a breakdown of what each part of the code does:

1.Getting Feature Names:

- **features = list(df.columns)[:-1]**: This line retrieves the names of the features (columns) from the DataFrame **df**, excluding the last column which is assumed to be the target variable.

2.Calculating Prior Probabilities:

- **prior = calculate_prior(df, Y)**: This line calculates the prior probabilities for each class in the target variable **Y** using the **calculate_prior** function, which is assumed to be defined elsewhere.

3.Iterating Over Data Samples:

- The function iterates over each data sample **x** in the input **X**.

4.Calculating Likelihood:

- Within the loop, it calculates the likelihood for each class using Gaussian distributions.
- It iterates over each class label **j** and each feature **i** to calculate the likelihood based on the Gaussian distribution parameters (mean and standard deviation) for each feature and class.

5.Calculating Posterior Probability:

- After calculating the likelihood for each class, it calculates the posterior probability (numerator only) by multiplying the likelihood with the corresponding prior probability for each class.

6.Predicting Class Labels:

- It predicts the class label for each data sample based on the class with the highest posterior probability.
- The predicted class labels are appended to the list **Y_pred**.

7.Returning Predictions:

- Finally, it returns an array of predicted class labels for all data samples.

Testing and accuracy :

Test Gaussian model

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=.2, random_state=41)

X_test = test.iloc[:,:-1].values
Y_test = test.iloc[:,-1].values
Y_pred = naive_bayes_gaussian(train, X=X_test, Y="diagnosis")

from sklearn.metrics import confusion_matrix, f1_score
print(confusion_matrix(Y_test, Y_pred))
print(f1_score(Y_test, Y_pred))
```

```
[[36  4]
 [ 0 74]]
0.9736842105263158
```

The accuracy : 0.9385964912280702

2)Categorical model:

A categorical model is used when the outcome variable is categorical, meaning it falls into distinct categories. It includes techniques like multinomial logistic regression, multinomial Naive Bayes, and decision trees [8].

```
def naive_bayes_categorical(df, X, Y):
    # get feature names
    features = list(df.columns)[:-1]

    # calculate prior
    prior = calculate_prior(df, Y)

    Y_pred = []
    # loop over every data sample
    for x in X:
        # calculate likelihood
        labels = sorted(list(df[Y].unique()))
        likelihood = [1]*len(labels)
        for j in range(len(labels)):
            for i in range(len(features)):
                likelihood[j] *= calculate_likelihood_categorical(df, features[i], x[i], Y, labels[j])

        # calculate posterior probability (numerator only)
        post_prob = [1]*len(labels)
        for j in range(len(labels)):
            post_prob[j] = likelihood[j] * prior[j]

        Y_pred.append(np.argmax(post_prob))

    return np.array(Y_pred)
```

Test and accuracy :

Test Categorical model

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=.2, random_state=41)

X_test = test.iloc[:,:-1].values
Y_test = test.iloc[:,-1].values
Y_pred = naive_bayes_categorical(train, X=X_test, Y="diagnosis")

from sklearn.metrics import confusion_matrix, f1_score
print(confusion_matrix(Y_test, Y_pred))
print(f1_score(Y_test, Y_pred))
```

```
[[38  2]
 [ 5 69]]
0.9517241379310345
```

Conclusion:

Naive Bayes classifiers are simple yet powerful tools for probabilistic classification tasks, particularly in scenarios involving text data and categorical features. Despite their "naive" assumption of feature independence, they often perform remarkably well in practice and find applications in various domains including text classification, spam filtering, healthcare, and marketing.

References:

[1] Lewis, D. (1998). Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In *Machine Learning: ECML-98*.

[2] Zhang, H. (2004). The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, pp. 562-567.

[3] McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, vol. 752, pp. 41-48.

[4] Zhang, H. (2004). The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, pp. 562-567.

[5] McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, vol. 752, pp. 41-48.

[6] Domingos, P., & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3), 103-130.

[7] Wasserman, L. (2004). *All of Statistics: A Concise Course in Statistical Inference*. Springer.

[8] Agresti, A. (2013). *Categorical Data Analysis*. John Wiley & Sons.