Naive Bayes

introduction to the problem and the Naive Bayes classifier.

The Naive Bayes classifier is a probabilistic machine learning model based on Bayes' Theorem. It is particularly suited for classification tasks where the feature space is high-dimensional.

$P(A|B)=(P(B)P(B|A))/P(A)$

In the context of the Titanic dataset, the Gaussian Naive Bayes classifier is appropriate since we have continuous features like Age and Fare. The process involves preprocessing the data (handling missing values, encoding categorical variables), training the model on a subset of the data, and then evaluating its performance on a test set.

Data exploration and preprocessing steps.

After looking at the data, I didn’t want some columns in the training or testing process.
["PassengerId","Name","Ticket","Cabin"]
Those columns will not affect anything in the process so I decided to drop them from my data frame.

Then I encoded the categorical columns like sex and embarked columns using map data structure.

Due to high null values in the age column, I decided to fill null values with mean instead of dropping them. dropping null values may lead to significant data loss.

Also, I dropped the duplicates from the dataframe.

Feature selection rationale.

The dataset you're working with is from the Titanic disaster, which is a classic dataset used in machine learning and data analysis. The goal is to predict whether a passenger survived the disaster based on various features such as age, sex, passenger class, fare, and other relevant information.

Model training process and parameter tuning.

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test,
y_pred)))
```

As shown in the previous code, all I have done is to choose the model, predict, and test the model. It got an accuracy 0.7564.

So I did the cross validation step to try to improve the accuracy as shown in the following code.

```
param_grid = {'var_smoothing': [1e-11, 1e-10, 1e-9, 1e-8, 1e-7, 1e-6]}
grid_search = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid,
cv=10, scoring='accuracy')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print(f'Best parameters found: {best_params}')
y_pred = grid_search.best_estimator_.predict(X_test)
print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test,
y_pred)))
```

But it got the same accuracy as before.