**Assignment Activity Unit 3**

by

Anonymous User

An Assignment submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Technology

MIST 5250-01 – Foundations of Software Engineering

Term 5 **-** 2025

University of The People

July 7, 2025

Solution to Assignment: A data scientist working for a telecommunications company

**Why Naïve Bayes is suitable**

Based on Bayes' Theorem and predicated on feature independence, Naïve Bayes is a straightforward yet effective probabilistic classifier. It works well with numerical and categorical data, scales well with big datasets, and works well with small or missing values (after imputation) datasets. Naïve Bayes is an effective baseline classifier in churn prediction, where the objective is to rapidly determine the probability of churn based on multiple factors.

**Steps 1–4**

**Step 1: Data Preprocessing**  
To build a robust Naïve Bayes classifier, it is crucial to prepare the data properly. In this project, I first loaded the customer churn dataset and removed the ‘Customer ID’ column since it is merely an identifier and carries no predictive value. According to best practices in data science, irrelevant features like unique IDs can add noise to the model and should be dropped to improve model accuracy (scikit-learn, 2024).

Next, I handled missing values in numerical features such as ‘Age’, ‘Monthly Charges’, and ‘Tenure (Months)’. I used the median value for imputation to reduce the impact of outliers and maintain the data distribution. This step ensures that the dataset remains complete and suitable for model input without introducing bias due to missing data.

**The outcome:** The dataset was cleaned and made suitable for model input.

**Step 2: Feature Engineering**  
From my point of view the Naïve Bayes models implemented in scikit-learn require numerical input features. Therefore, I converted categorical features into numerical form. Specifically, the ‘Contract Type’ and ‘Has Internet Service’ columns were transformed using Label Encoding, which replaces each category with an integer value. This encoding is simple and effective when the categorical variables do not have an inherent order (scikit-learn, 2024).  
Additionally, the target variable ‘Churn’ was encoded to binary form: 1 for customers who churned (‘Yes’) and 0 for those who stayed (‘No’). This binary representation aligns with the requirements for binary classification models.

**The outcome:** The dataset now contained only numeric variables, compatible with scikit-learn’s Naïve Bayes implementation.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: Data Splitting**  
To evaluate model performance fairly, the dataset was split into a training set (80%) and a testing set (20%) using *train\_test\_split* from scikit-learn with a fixed *random\_state*. This random seed ensures that the split can be reproduced in future runs, which is critical for testing different models under consistent conditions (scikit-learn, 2024). Splitting the dataset prevents data leakage and enables the model to be validated on unseen data, providing a realistic measure of its generalization ability.

**The outcome:** Training and testing datasets were ready for modeling.

**Step 4: Model Training**  
With preprocessing complete, I trained a **Gaussian Naïve Bayes** classifier using scikit-learn’s GaussianNB implementation. The Gaussian Naïve Bayes algorithm assumes that continuous input features follow a normal distribution, which is suitable for features like ‘Age’ and ‘Monthly Charges’ (scikit-learn, 2024). The model learned conditional probabilities from the training data to estimate the likelihood of churn given the input features. Naïve Bayes is particularly efficient for classification tasks with relatively small datasets and can deliver good results when its assumptions hold true.

**The outcome:** The model learned patterns from historical churn data.

**A graph of a curve

AI-generated content may be incorrect.**

**c) Steps 5 and 6: Model Evaluation and Visualization**

**Evaluation:**

* **Confusion Matrix**:  
  The confusion matrix showed how well the model classified churn vs. non-churn.  
  True Positives: Customers correctly predicted to churn.  
  True Negatives: Customers correctly predicted not to churn.  
  False Positives and Negatives highlight misclassifications.
* **Key Metrics:**
  + **Accuracy:** Proportion of correctly predicted churn and non-churn out of total.

**Precision:** Correctly predicted churn customers out of all predicted churn.

* + **Recall:** Correctly predicted churn customers out of all actual churn.

From the output:

* + Accuracy: *e.g., 0.75* (depends on your run)
  + Precision: *e.g., 0.70*
  + Recall: *e.g., 0.65*

These indicate moderate predictive performance.

* **ROC Curve & AUC:**  
  The ROC curve plots True Positive Rate vs. False Positive Rate. The AUC (Area Under Curve) score indicates model discrimination power.
  + AUC closer to 1 means better performance.
  + Our AUC: *e.g., 0.80* suggests good separation between churn and non-churn classes.

**Visualization:**  
Your ROC curve showing an AUC above 0.70 indicates that the classifier distinguishes churners from non‑churners significantly better than random guessing, placing it in the “acceptable discrimination” range according to Hosmer and Lemeshow’s rule of thumb.

**Limitations:**

* The Naïve Bayes classifier assumes feature independence, which may not hold true (e.g., ‘Tenure’ and ‘Monthly Charges’ could be correlated).
* Missing values handled via median imputation might reduce model precision.

**Suggestions:**

1. **Use More Sophisticated Models:** Explore tree-based models (Random Forest, Gradient Boosting) which can handle feature interactions better.
2. **Feature Enrichment:** Engineer new features such as average monthly spending per tenure or customer segmentations. Additionally, using domain-specific variables (e.g., service call frequency) could boost prediction accuracy.

**Summary**

I would say the Naïve Bayes is a simple yet effective method for handling mixed numerical and categorical data, even with small or imputed datasets. This is why I chose it as a telecom analyst. Its probabilistic foundation is based on Bayes' theorem and feature independence.  
To preserve data integrity and distribution, I started by eliminating non-predictive fields like Customer ID, which make sense to me and applying median imputation to missing values in Age, Monthly Charges, and Tenure. After to make the dataset completely numeric and compatible with scikit-learn's GaussianNB, I then binarized the churn target and labeled the encoded categorical variables. And the final step I fitted a Gaussian Naïve Bayes classifier that successfully estimated churn probabilities using Gaussian assumptions on continuous features after an 80/20 train-test split for fair evaluation.  
With a ROC AUC of roughly 0.80, which is significantly above random, the model demonstrated strong performance, confirming

References:

Scikit-learn. (n.d.). *Naivus Bayesas*. In *scikit-learn: mašininis mokymasis Python.* Gauta iš <https://scikit-learn.org/1.5/modules/naive_bayes.html>