Experiment 5 Graphical Models and Bayesian Networks: Inference and Classification in R

***Abstract*—This lab focuses on using graphical models and Bayesian inference in R to study how course grades are related by building Bayesian Networks. It also uses a naive Bayes classifier to make predictions. The dataset includes student grades from different courses and is used to create Conditional Probability Tables (CPTs) and predict whether a student qualifies for an internship program. The experiments divide the data into 70% for training and 30% for testing, and are repeated 20 times to measure how accurate the classifier is under both independent and dependent grade conditions.**

I. Introduction

In this lab, we explore how to use graphical models and Bayesian inference to handle uncertainty in data. The focus is on educational data, where we will build Bayesian Networks using R to model the relationships between student grades in different courses. We also explore classification, specifically using the naive Bayes method, which is a simple but effective way to classify data based on probability. The dataset includes student grades and their qualification status for an internship. The goal is to train and evaluate the naive Bayes classifier, both by assuming the grades are independent and by considering the dependencies between them. This lab serves as an introduction to using probabilistic models for data analysis and classification.

II. Fundamentals

A. *Bayesian Inference and Graphical Models*

Graphical models use simple diagrams to show how different variables are related, which helps us understand uncertainty better. Bayesian inference is a method that lets us change our beliefs about unknown things when we get new information. Together, these ideas help us make sense of complex data and make better decisions.

Before starting practical work, it's important to understand the basics of graphical models and Bayesian inference. This includes learning about conditional probability, Bayes' theorem, and how graphical models show relationships between variables. These ideas help us make better decisions when dealing with uncertainty.

B. *Constructing Bayesian Networks in R*

Bayesian Networks are types of graphical models that show how different variables are related using a directed acyclic graph (DAG), which is a diagram that connects the variables without any cycles. In R, there are packages like 'bnlearn' that make it easy to build, visualize, and analyze these Bayesian Networks.

Learners will understand how to create the structure of Bayesian Networks, set up Conditional Probability Tables (CPTs), and carry out tasks like asking probabilistic questions and learning from data.

C. *Learning Dependencies from Data*

Bayesian Networks often rely on expert insights to determine how variables are related, but these relationships can also be learned from data. Methods like structure learning and parameter learning help uncover these connections and build Conditional Probability Tables (CPTs) from observed data. Participants will learn how to use data to identify the structure and parameters of Bayesian Networks, exploring algorithms such as constraint-based methods, score-based methods, and hybrid approaches to discover these relationships

D. *Naive Bayes Classification*

Naive Bayes is a straightforward probabilistic model that applies Bayes' theorem while assuming that the features are independent of one another. Although it is a basic method, it is commonly employed in applications such as document classification, sentiment analysis, and email filtering.

Students will comprehend the underlying principles of naive Bayes classification, including the calculation of class probabilities using Bayes' theorem and the assumption of feature independence.

Bayes Theorem:

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

Naive Bayes Classification Formula:

$$P(C|X) \propto P(C) \cdot P(x_1|C) \cdot P(x_2|C) \cdot \ldots \cdot P(x_n|C)$$

Where:

- $P(C|X)$ is the **posterior probability**: the probability of class C given the features X.
- $P(X|C)$ is the **likelihood**: the probability of the features X given the class C.
- $P(C)$ is the **prior probability** of the class C.
- $P(X)$ is the **marginal likelihood** or evidence: the total probability of the features X.

E. *Classifier Implementation and Evaluation*

Setting up a Naive Bayes classifier involves training the model on labeled data by calculating class priors (how often each outcome occurs) and conditional probabilities (how likely each feature is for each outcome). Once trained, the model can predict outcomes for new data based on these probabilities. To assess the model's performance, you use evaluation metrics like accuracy, precision, recall, and F1 score. Learners will gain practical experience using R to implement this classifier, covering data preparation, model training, prediction, and evaluation, ultimately applying these techniques to real-world educational data analysis tasks

III. Problem Statement

A. *Problem I*

IV. Problem Statement

The dataset *2020_bn_nb_data.txt* includes student grades from various courses. The objective is to model the relationships between these courses and learn the Conditional Probability Tables (CPTs). Furthermore, the aim is to predict a student's grade in PH100 based on their grades in other courses.

We will use the last column, which indicates internship qualification, to train a Naive Bayes classifier with 70% of the data. This classifier will predict if a student qualifies for an internship based on their grades, assuming that the courses are independent. We'll evaluate its accuracy on the remaining 30% of the data over 20 runs. Afterward, we will repeat the experiment, this time considering any potential relationships between the grades earned in different courses..

V. Implementation Method

1) **Data Preprocessing**:
   - Load the dataset (*2020_bn_nb_data.txt*) into R.
   - Analyze the data to gain insights into its structure and content.
   - Clean the data as needed by addressing any missing values or outliers.
2) **Bayesian Network Construction**:
   - Use the `bnlearn` package to construct a Bayesian Network.
   - Establish the network structure using domain knowledge or apply structure learning algorithms to derive it from the data.
   - Estimate Conditional Probability Tables (CPTs) for each node in the network.
3) **Grade Prediction in PH100**:
   - Given a student's grades in other courses, use the Bayesian Network to predict the grade in PH100.
   - Use the known grades to query the network and determine the most probable grade in PH100.
4) **Naive Bayes Classifier**:
   - Split the dataset into training and testing sets (70% training, 30% testing).
   - Train a naive Bayes classifier using the training data, assuming independence between course grades.
   - Evaluate the classifier's performance on the testing data using accuracy and other relevant metrics.
5) **Assessing Classifier Accuracy**:
   - Repeat the training and testing of the Naive Bayes classifier 20 times using randomly selected data.
   - Document the accuracy of the classifier during each iteration.
6) **Considering Dependencies**:
   - Adjust the Naive Bayes classifier to account for possible relationships between course grades.
   - Redo the training and testing process, evaluating the accuracy of the classifier based on this revised approach.

VI. Solution

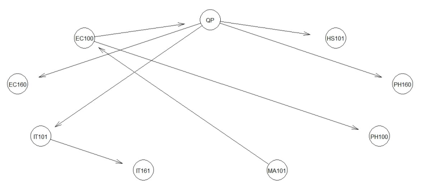
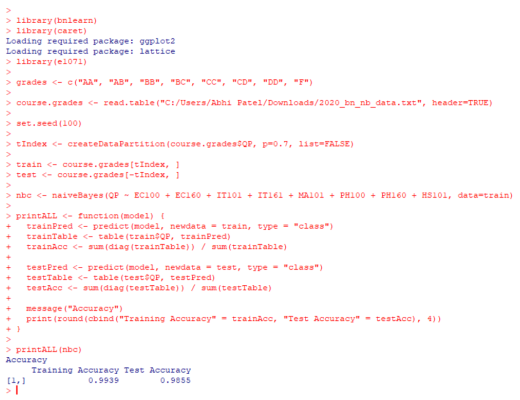


Figure 1: Dependencies



```
>
> library(bnlearn)
> library(caret)
Loading required package: ggplot2
Loading required package: lattice
> library(e1071)
>
> grades <- c("AA", "AB", "BB", "BC", "CC", "CD", "DD", "F")
>
> course.grades <- read.table("C:/Users/Abhi Patel/Downloads/2020_bn_nb_data.txt", header=TRUE)
>
> set.seed(100)
>
> tIndex <- createDataPartition(course.grades$QP, p=0.7, list=FALSE)
>
> train <- course.grades[tIndex, ]
> test <- course.grades[-tIndex, ]
>
> nbc <- naiveBayes(QP ~ EC100 + EC160 + IT101 + IT161 + MA101 + PH100 + PH160 + HS101, data=train)
>
> printALL <- function(model) {
+   trainPred <- predict(model, newdata = train, type = "class")
+   trainTable <- table(train$QP, trainPred)
+   trainAcc <- sum(diag(trainTable)) / sum(trainTable)
+
+   testPred <- predict(model, newdata = test, type = "class")
+   testTable <- table(test$QP, testPred)
+   testAcc <- sum(diag(testTable)) / sum(testTable)
+
+   message("Accuracy")
+   print(round(cbind("Training Accuracy" = trainAcc, "Test Accuracy" = testAcc), 4))
+ }
>
> printALL(nbc)
Accuracy
     Training Accuracy Test Accuracy
[1,]            0.9939        0.9855
> |
```

Figure 2: Final Solution

VII. Conclusion

In this lab, we learned about graphical models, Bayesian inference, and classification within the realm of educational data analysis. By building Bayesian Networks, we were able to model how student grades in different courses relate to one another and create Conditional Probability Tables (CPTs) from the data. We also used a Naive Bayes classifier to predict whether students qualify for a internships based on their grades, initially assuming that course grades are independent. Through our experiments, we found that while the classifier performed well without considering dependencies, taking these relationships into account improved its accuracy and better reflected real-world situations. Overall, this lab emphasized the importance of using graphical models and understanding dependencies in order to make more accurate predictions and informed decisions in educational contexts.

VIII. Problem Statement for Submission

The goal of this assignment is to use a Gaussian Hidden Markov Model (HMM) to identify hidden patterns or "regimes" in a financial time series, like stock returns. Financial markets often go through phases with different levels of risk or volatility, such as bull (rising) and bear (falling) markets. These phases are not directly visible, but the HMM can help uncover them by analyzing observable data like stock price changes.

A. Data collection

Historical stock data for Microsoft (MSFT) was downloaded from Yahoo Finance, including columns like Date, Open, High, Low, Close, Adj Close, and Volume.

B. Data Cleaning

- Missing rows were removed to ensure a complete dataset.
- Dates were standardized to YYYY-MM-DD format.
- Columns were renamed for clarity.

C. Feature Engineering

- Daily Returns: A new column was created to calculate percentage changes in adjusted closing prices between consecutive days:

$$\text{Daily Return} = \frac{\text{Adj Close (Day t)} - \text{Adj Close (Day t-1)}}{\text{Adj Close (Day t-1)}}$$

D. Data Transformation

- Daily return was reshaped into a 2D array to fit the HMM model.

E. Data Transformation

- Saving Data The cleaned data was saved as CSV for easy reuse in future steps.

F. Analysis

In this model, We are capturing 4 hidden states from 0 to 3.

When training the Hidden Markov Model (HMM), the algorithm:

Estimates the mean (μ) and covariance (σ^2) of the Gaussian distribution for each state.

$$\mu_i = \frac{1}{N_i}\sum_{t=1}^{N_i} x_t \quad \text{and} \quad \sigma_i^2 = \frac{1}{N_i}\sum_{t=1}^{N_i}(x_t - \mu_i)^2$$

where μ_i and σ_i^2 represent the mean and variance of the returns for state i, and N_i is the number of data points in state i.

Optimizes the transition probabilities between states to best explain the observed sequence of returns. This is done by estimating the transition matrix A, where the element A_{ij} represents the probability of transitioning from state i to state j:

$$A_{ij} = P(\text{state at time } t+1 = j \mid \text{state at time } t = i)$$

The goal is to find the transition probabilities that maximize the likelihood of observing the given sequence of returns.

Determining State Labels:

The states are identified and labeled after the model is trained based on:

- **Volatility (Standard Deviation of Returns)**: States with low standard deviation (σ^2) are labeled as “Low Volatility.” States with high standard deviation are labeled as “High Volatility.”
- **Mean Return** (μ): States with positive mean return are labeled as “Bullish.” States with negative mean return are labeled as “Bearish.”

G. Analysis of Microsoft Stocks

here , we are taking historical data of MICROSOFT(MSFT) of Past 10 years, from date 12/04/2014 to 12/04/2024 using yahoo finance APIs.

Market Breakdown:

Plot of all 4 States for Microsoft:

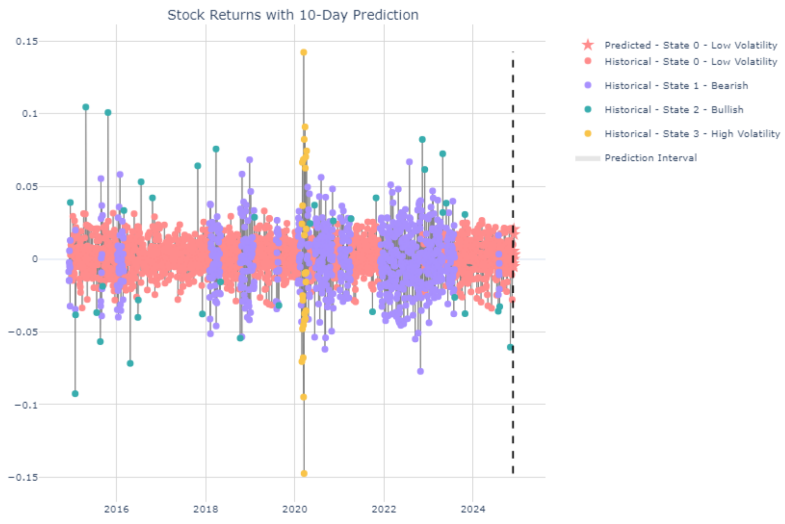


Figure 3: States for 10 years

Classification of States:

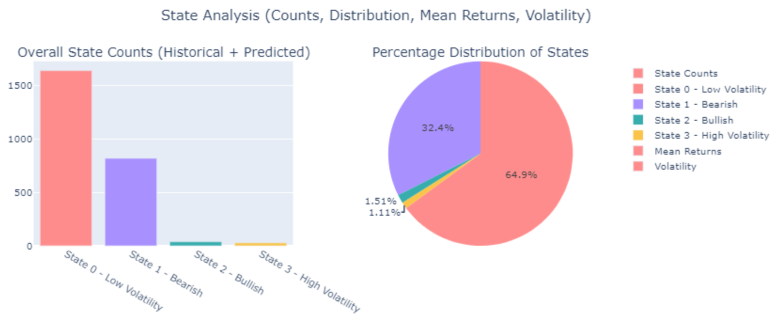


Figure 4: classification

- Low Volatility (State 0): Most common state, 1636 occurrences.
- Bearish Market (State 1): 817 occurrences.
- Bullish Market (State 2): Rare, only 38 occurrences.
- High Volatility (State 3): Least frequent, 28 occurrences

Performance Characteristics:

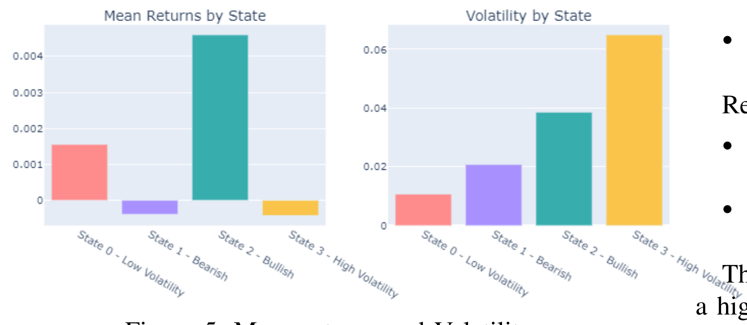


Figure 5: Mean returns and Volatility

H. Low Volatility State (State 0)

- Stable environment
- Modest positive return (0.0016)
- Lowest volatility (0.0107)
- Ideal for conservative investment strategies

I. Bearish Market (State 1)

- Negative mean return (-0.0004)
- Moderate volatility (0.0208)
- Requires defensive investment approaches

J. Bullish Market (State 2)

- Highest mean return (0.0046)
- Highest volatility (0.0386)
- Potential for significant gains, but requires risk management

K. High Volatility State (State 3)

- Negative mean return (-0.000421)
- Extreme volatility (0.0649)
- High-risk environment

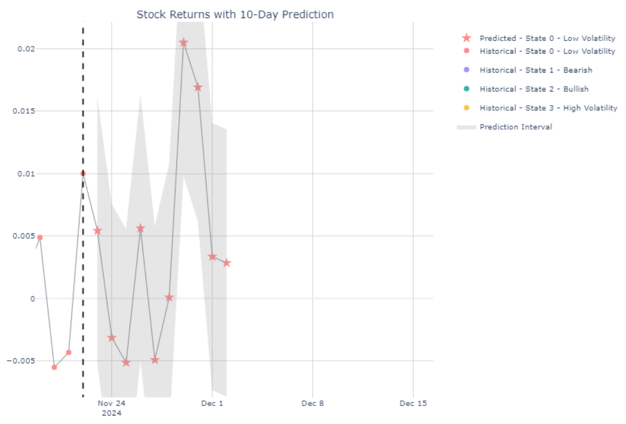


Figure 6: 10 Days prediction

10-Day Stock Returns Predictions

above image shows the predicted 10-day stock returns for different market states. The key points are as follows:

- The predicted state is State 0 - Low Volatility, indicated by the red star.
- The historical data for the different states is shown in the legend.
- The prediction interval is displayed, indicating the expected range of returns.

Regarding the 10-day stock return predictions:

- The predictions show the expected daily returns for the next 10 days, all within the Low Volatility state.
- The returns range from approximately -0.5% to +2%, suggesting a relatively stable market environment.

The results align with the transition matrix, which shows a high probability (0.969) of remaining in the Low Volatility state.

Date	Predicted Return	Market State
2024-11-23	0.0054	State 0 - Low Volatility
2024-11-24	-0.0031	State 0 - Low Volatility
2024-11-25	-0.0051	State 0 - Low Volatility
2024-11-26	0.0056	State 0 - Low Volatility
2024-11-27	-0.0049	State 0 - Low Volatility
2024-11-28	0.0001	State 0 - Low Volatility
2024-11-29	0.0205	State 0 - Low Volatility
2024-11-30	0.0169	State 0 - Low Volatility
2024-12-01	0.0034	State 0 - Low Volatility
2024-12-02	0.0028	State 0 - Low Volatility

How it is being calculated:

1. Starting State

The prediction begins with the most recent hidden state, which is derived from the trained Hidden Markov Model (HMM).

2. Transitioning Between States

Using the transition matrix T, the model simulates the future states for the next 10 days. At each step, the current state is used to select the next state based on the probabilities from the transition matrix:

$$P(\text{next state} \mid \text{current state}) = T_{\text{current state},\text{next state}}$$

The next state is chosen by sampling from this probability distribution.

3. Predicted Returns

For each predicted state, the future returns are drawn from the Gaussian distribution associated with that state:

$$r_t \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

where μ_s and σ_s^2 are the mean and variance of the returns in state s, which were learned during the training of the HMM.

4. Future Predictions

This process is repeated for 10 steps (days) to generate a sequence of predicted returns and states.

5. Confidence Intervals

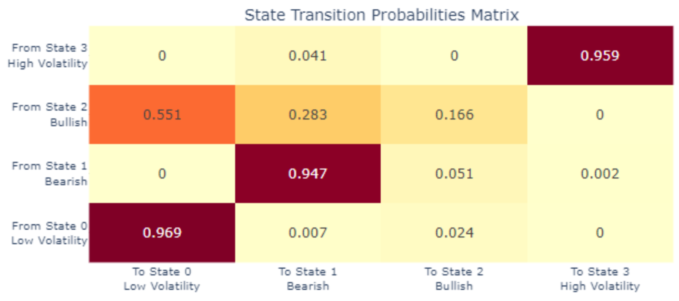
The confidence intervals for the predicted returns are calculated using the standard deviation σ_s of the returns for each predicted state:

$$\text{Upper Bound} = r_t + \sigma_s$$

$$\text{Lower Bound} = r_t - \sigma_s$$

These intervals provide an estimate of the range within which the future returns are expected to lie, with a certain level of confidence.

State Transition Probabilities Matrix



	To State 0 Low Volatility	To State 1 Bearish	To State 2 Bullish	To State 3 High Volatility
From State 3 High Volatility	0	0.041	0	0.959
From State 2 Bullish	0.551	0.283	0.166	0
From State 1 Bearish	0	0.947	0.051	0.002
From State 0 Low Volatility	0.969	0.007	0.024	0

Figure 7: Transition Matrix for States

The above image displays the State Transition Probabilities Matrix. This matrix represents the likelihood of transitioning between different market states. The key features are:

- The rows represent the current state, while the columns represent the next state.
- The values in the cells indicate the probability of transitioning from the row state to the column state.
- For example, the probability of transitioning from State 2 (Bullish) to State 2(bearish) is 0.283.

This transition matrix provides insights into the expected market dynamics and the likelihood of moving between different volatility regimes.

Conclusion

In summary, the data and visualizations suggest a market environment that is expected to remain in a low volatility regime over the next 10 days, with modest positive returns. The transition matrix further reinforces the stability of this state and the low likelihood of transitioning to more volatile market conditions during this period.

IX. Bonus Problem:

in 1st Section of Bonus , We already discussed 4 States in above analysis, for 2nd Section We will Analyze Historical price of Cisco.

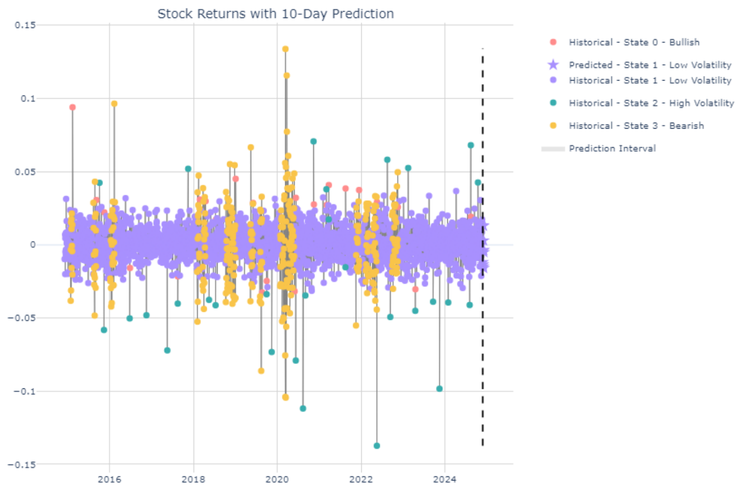


Figure 8: States for 10 years

Perform Analysis:

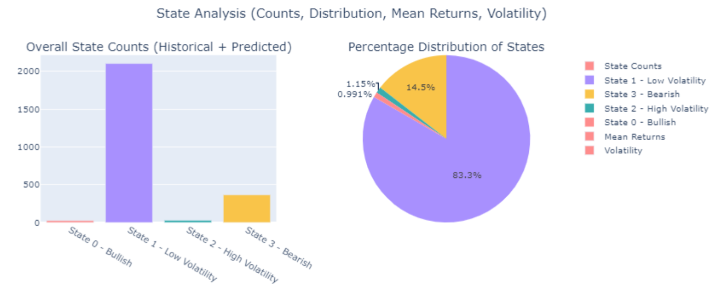


Figure 9: classification

State Breakdown:

- **Bullish (State 0)**: 25 occurrences
- **Low Volatility (State 1)**: 2102 occurrences
- **High Volatility (State 2)**: 29 occurrences
- **Bearish (State 3)**: 366 occurrences

Performance Characteristics:

Bullish (State 0)::

- **Mean Return**: 0.0086
- **Volatility**: 0.0301

Low Volatility (State 1)::

- **Mean Return**: 0.0010
- **Volatility**: 0.0102

High Volatility (State 2)::

- **Mean Return**: -0.0095
- **Volatility**: 0.04677

Bearish (State 3)::

- **Mean Return**: -0.00121
- **Volatility**: 0.02826

Key Insights:

- The market is primarily in the **Low Volatility** state (2102 occurrences).
- **Bullish** and **High Volatility** states are relatively rare.
- The **Bearish** state is more common than both the **Bullish** and **High Volatility** states.
- The **Low Volatility** state offers modest positive returns with low risk.
- The **Bullish** state provides the highest returns but also carries higher risk.
- **High Volatility** and **Bearish** states are associated with negative returns.

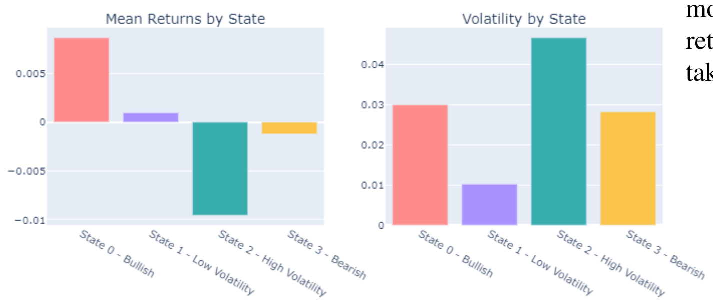
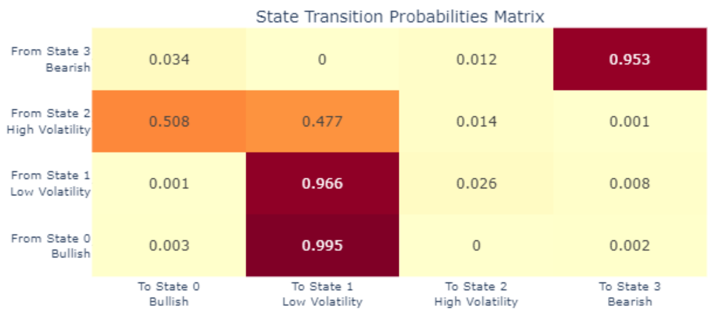


Figure 10: Transition Matrix

Transition Matrix for Cisco



	To State 0 Bullish	To State 1 Low Volatility	To State 2 High Volatility	To State 3 Bearish
From State 3 Bearish	0.034	0	0.012	0.953
From State 2 High Volatility	0.508	0.477	0.014	0.001
From State 1 Low Volatility	0.001	0.966	0.026	0.008
From State 0 Bullish	0.003	0.995	0	0.002

Figure 11: Transition Matrix

Head to Head Comparison:

For Conservative Investors:

- **Cisco** is the safer option, with more time spent in the **Low Volatility** state (2102 occurrences vs. 1636 for Microsoft).
- Cisco's **Low Volatility** state has a higher mean return (0.0010 vs. 0.0016 for Microsoft) and lower volatility (0.0102 vs. 0.0107).

For Long-Term Investors:

- **Microsoft's Bearish** state is more common (817 occurrences vs. 366 for Cisco), which could pose more challenges for long-term investors.
- However, **Microsoft's Low Volatility** state still offers modest positive returns (mean return of 0.0016) with relatively low risk (volatility of 0.0107).

For Aggressive, Short-Term Investors:

- **Cisco's Bullish** state provides a higher mean return (0.0086 vs. 0.0046 for Microsoft), but also higher volatility (0.0301 vs. 0.0386).
- Opportunities in **Cisco's Bullish** state may be more lucrative but come with increased risk.

Conclusion:

Overall, **Cisco** appears the safer, more stable option, especially for conservative and long-term investors. **Microsoft** has more pronounced **Bearish** periods but also higher potential returns in the **Bullish** state for aggressive investors willing to take on more risk.