DSE 6311 – Data Science Capstone

Team Gamma

**Team Lead:** Crozier, Amber

croziera@merrimack.edu

**Reporter:** Canfield, Ryan

canfieldr@merrimack.edu

**Spokesperson:** Sabri, Abdelmalek

sabria@merrimack.edu

EVALUATING AI’s ROLE IN CUSTOMER SATISFACTION AND RETENTION

Initial Model and Tuning Report

Table of Contents

[**Background & Question 2**](#_gjdgxs)

[Question: 2](#_30j0zll)

[Hypothesis and Prediction 2](#_3dy6vkm)

[Hypothesis: 2](#_1t3h5sf)

[Prediction: 2](#_4d34og8)

[**Introduction 2**](#_2s8eyo1)

[**Methods 2**](#_m31njub75k71)

[Initial Model: 2](#_17dp8vu)

[Justification: 2](#_26in1rg)

[Cross-validation: 3](#_lnxbz9)

[Model Assumptions: 3](#_3rdcrjn)

[Tuning and Additional Models: 3](#_uskc121jr9rk)

[Overfitting: 3](#_rxtn7j3y1qn5)

[Hyperparameter Tuning: 3](#_z819g2o3jt21)

[**Results 4**](#_35nkun2)

[Initial Model: 4](#_8c1h5b75v0y5)

[Analysis of Algorithm: 4](#_st91uhfz290s)

[Analyzing Assumptions: 5](#_73os8yi78htt)

[Plans for Overfitting: 6](#_53dbkyoldml9)

[Tuning Model: 6](#_op41an79odzp)

[Cross-validation and Hyperparameter Tuning: 6](#_a278z2jy2pjc)

[Evaluation: 6](#_5ktez3v2tz0c)

[**Discussion and Next Steps 9**](#_ig86pdrxqrst)

[Summary of Key Takeaways: 9](#_oevtynuhanj9)

[Further Tuning Plan: 9](#_37lgxar2zkt3)

[**GitHub Link 9**](#_k3pl36efu8r5)

[**References 10**](#_2xcytpi)

[**Appendix 11**](#_m3adjyvj41xm)

# Background & Question

## Question:

Can we predict customer satisfaction with AI content to help businesses identify the type of customer base most likely to have a positive experience with AI tools they deploy?

## Hypothesis and Prediction

### Hypothesis:

While the implementation of AI in retail settings improves cost efficiency and reduces errors (Kasareni, 2021), excessive reliance on AI in customer service without human interaction will reduce customer satisfaction.

### Prediction:

Customer satisfaction with AI-driven customer service will be higher among specific demographic groups and for certain AI tools.

# Introduction

Last year in 2023 according to Bloomberg, big tech companies have been spending over $200 Billion dollars on AI. To this day AI use is continually increasing within the business model and it’s important to identify which customer segments respond positively to AI and their preferred AI tools. This project focuses on predicting customer satisfaction based on demographic and usage data, with the aim of helping businesses identify customer segments most likely to have a positive experience with AI tools. After performing exploratory data analysis (EDA), which identified patterns in demographic features and AI usage, we chose to use a neural network model for its flexibility in handling complex relationships. PCA was used to simplify the features while keeping important information for prediction. This project aims to provide insight into when and why satisfaction with AI decreases in order to create actionable insights for businesses to implement AI efficiently.

# Methods

After conducting exploratory data analysis (EDA) and identifying key features we selected a deep learning model to capture the non-linear relationships between these features and customer satisfaction. The choice of a neural network model was based on its ability to handle categorical data effectively, and its capacity to model complex interactions between features. We also used Component Analysis (PCA) to reduce dimensionality and to reduce feature space and address overfitting.

## Initial Model:

### Justification:

For the initial model, we implemented a deep learning model using Keras with three dense layers, each followed by a ReLU activation function, and a final output layer with a sigmoid activation function to predict a binary outcome (satisfied vs. not satisfied). This was selected because neural networks are ideal for learning non-linear relationships. Part of our assumptions were that the relationship between the features was not linear and was more complex. We used the RMSProp optimizer to train the model because we wanted to reduce noise and we used binary cross-entropy as the loss function because this project was a binary classification problem.

### Cross-validation:

We employed cross-validation using an early stopping mechanism with a patience value of 10 to prevent overfitting and ensure that the model generalized well. The early stopping monitored the validation loss, and ran 10 consecutive epochs to see if there was any improvement. When improvement was seen training was halted and best model weights were restored. This ensured that the model didn’t continue training beyond the point where it began to overfit.

### Model Assumptions:

The assumptions of the model included the expectation that the relationships between features and customer satisfaction are non-linear and that PCA would keep enough variance in the dataset to be able to see important patterns to predict. We also assumed that the features were independent, this assumption seemed reasonable based on the exploratory data analysis (EDA).

## Tuning and Additional Models:

### Overfitting:

To address overfitting, we included dropout layers in both the initial and the PCA mode to reduce the model's reliance on specific neurons. In the PCA model, we also added an additional dropout layer to enhance regularization. The training was also monitored using early stopping to prevent unnecessary epochs once the validation loss stopped improving. Early stopping ensured the model did not train excessively, which could have led to overfitting. The batch size of 512 and the number of epochs set to 100 were chosen to balance training time and model generalization, as large batch sizes and too many epochs could have led to overfitting by making the model too specific to the training data. PCA was also used to reduce the dimensionality of the dataset to make sure the model was not too complex.

### Hyperparameter Tuning:

For hyperparameter tuning, we started by splitting the dataset into training and test sets. We optimized the split at 78% in the training set and held out 22% for the test set. We then optimized the neural network’s architecture and training settings by manually selecting the number of hidden layers and neurons in each layer (4 hidden layers with 256, 124, 64, and 12 neurons) based on previous experience and model experimentation. A dropout rate of 0.2 was chosen to reduce overfitting without significantly inhibiting the model’s ability to learn. The batch size of 512 was chosen because it was large enough to efficiently process data without causing memory issues. The number of epochs was set to 100, which balanced the training duration with the risk of overfitting. The RMSprop optimizer was selected for its ability to handle sparse gradients and adaptive learning rates, which helped in faster convergence and stabilized the training process.

# Results

## Initial Model:

### Analysis of Algorithm:

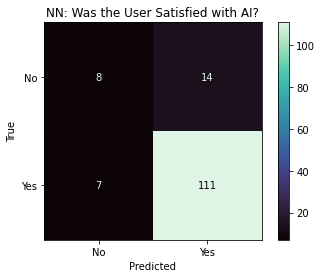
Upon analyzing the initial neural network model and every complete pass the neural network did, it revealed many things to us. If we look at the first run or epoch the model completed, it showed us that the model has a promising accuracy of 82% with a loss of 0.6. When testing the model on the validation data, the model had a validation accuracy of 82% and a validation loss of 0.52. By the last or 100th completed pass, the neural network had an accuracy of 99.7% with a loss of 0.78. It also got a validation accuracy of 78% and a validation loss of 1.13. Below is a table to summarize the information above which compares the first and final pass that the Neural Network made and a graph that shows accuracy across all epochs.

| Epoch | 1 | 100 |
| --- | --- | --- |
| Accuracy | 0.8152 | 0.9970 |
| Loss | 0.5933 | 0.0346 |
| Validation Accuracy | 0.8171 | 0.7805 |
| Validation Loss | 0.5213 | 1.1283 |

A graph of a graph showing a line of a graph

Description automatically generated with medium confidence

Loss is a metric that shows how well the model's predictions match the actual target values. The goal of a good model is to have a low loss value since the lower the loss value the more accurate the model will be. So even though our model is increasing its accuracy and decreasing loss over the 100 runs, it has also decreased its validation accuracy and validation loss every run which is a sign of overfitting. The reason why I think the model is overfitting is because it is making some general assumptions on the data. Even though the model seems to be overfitting, here are the evaluation metrics after the model predicted on the test data. A confusion matrix of the test results showed that the model achieved an accuracy of 85%, recall of 94.07%, precision of 88.8%, specificity of 36.36%, and an F1 score of 91.36%. The accuracy indicates that 85% of the model’s predictions were correct, but the lower specificity suggests that the model has a higher rate of false positives, meaning it incorrectly classifies customers who are not satisfied as satisfied. On the other hand, the high recall shows that the model is effective at detecting most of the instances where a customer is satisfied. With all these base metrics we can still improve the model by dealing with some of its assumptions and fine tuning the hyper parameters.



|  |  |
| --- | --- |
| **Evaluation Metrics** |  |
| Accuracy | **85.00 %** |
| Recall | **94.07 %** |
| Precision | **88.80 %** |
| Specificity | **36.36 %** |
| F1 | **91.36 %** |

### 

### Analyzing Assumptions:

The first assumption is that our model thinks in a non linear way since we decided to use a “relu” activation function. Just because we are using this does not mean the data isn’t linear. The next assumption is that there is enough data for each run for the model to learn something new. Our dataset is relatively small with it only having about 600 rows and 22 columns. That number gets cut down when it is split into a training set and a validation set. When the model runs through an epoch the model goes through the data and tries to learn something new even though it might skew the accuracy of the predictions. The features included could also have high multicollinearity since we included all and did not drop any of the features that were similar. Another assumption that it could be making is that it assumes that the training data is independently and identically distributed ([IID](https://medium.com/@evertongomede/independent-and-identically-distributed-iid-in-machine-learning-assumptions-and-implications-930ee9821e14)), meaning that each sample of data comes from the same source and is not influenced by the other samples. Even though this is not always true in the real world when the data was collected, NN models can often jump to this conclusion.

### Plans for Overfitting:

To control the model’s overfitting we added PCA to the data. PCA will reduce the number of features in our dataset and only capture the most important information and patterns from our set of features. Next, we plan to add cross validation to each epoch run. This will be used to evaluate the model's performance each run by splitting the data into multiple subsets, and checking the trained model each time making sure it does not overfit. We will also add dropout, which will randomly dropout or deactivate a certain percentage of neurons during each layer of the NN step. This will force the model to create new patterns and prevent it from becoming too reliant on specific neurons. Finally we will add an early stoppage. Early stoppage makes the model stop running through another epoch if the validation loss starts to increase and then keeps increasing.

## Tuning Model:

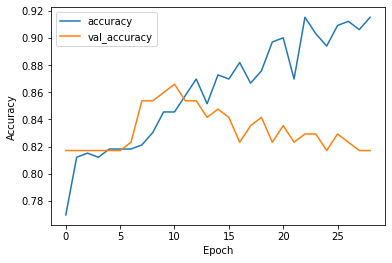
### Cross-validation and Hyperparameter Tuning:

As previously mentioned we were able to employ all of the cross validation and hyperparameter tuning for our NN model. After every epoch run our model compared the partially trained model on validation data. Using a search grid function and trying different activation, optimizer, and loss functions the best was still “relu” for activation, “rmsprop” for the optimizer, and “binary\_crossentropy” for the loss function. We also perform PCA and take all the principal components that explained 95% of the variance. Finally to deal with the overfitting we added early stoppage and dropout. For the early stoppage we gave it a patience of 20 and dropped out 20% of the neurons on each layer. This dropped the number epochs down from 100 to 29 and only stopped when the validation loss started to increase.

### Evaluation:

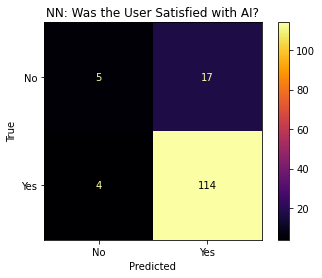
We are using the same metrics to evaluate the overfitting and the model’s overall improvement. For the overfitting we looked at the new training accuracy and validation loss. From the first epoch of the new neural network model we got an accuracy of 0.7697 and a validation loss of 0.5254. On the last epoch of this new model which is number 29 we got an accuracy of 0.9152 and a validation loss of 0.4540. The table below summarizes these new values as well as compares the previous values from the first model. There is also a plot showing the new model's accuracy across all of the training.

|  | New Model | New Model | Old Model | Old Model |
| --- | --- | --- | --- | --- |
| Epoch | 1 | 29 | 1 | 100 |
| Accuracy | 0.7697 | 0.9152 | 0.8152 | 0.9970 |
| Loss | 0.6017 | 0.2416 | 0.5933 | 0.0346 |
| Validation Accuracy | 0.8171 | 0.8171 | 0.8171 | 0.7805 |
| Validation Loss | 0.5254 | 0.4540 | 0.5213 | 1.1283 |



As you can see from the table and graph above the model is overfitting less then the initial model. Even though the new model has a slightly lower accuracy than the initial model it has a greatly reduced validation loss value. For the model’s improvement we can look at the confusion matrix and how it did on the test data.

|  |  |  |
| --- | --- | --- |
| **Evaluation Metrics** | Initial Model | Improved Model |
| Accuracy | **85.00 %** | **85.00 %** |
| Recall | **94.07 %** | **96.61 %** |
| Precision | **88.80 %** | **87.02 %** |
| Specificity | **36.36 %** | **22.73 %** |
| F1 | **91.36 %** | **91.57%** |



Even though the initial model performed well, there are noticeable improvements in the evaluation metrics after all the adjustments were made. The accuracy of the model remained the same at 85% for both the initial and improved models. This can suggest that the overall correct classification rate didn’t change. However, the recall improved from 94.07% to 96.61%, showing that the improved model is now better at detecting positive instances, meaning it’s more effective at identifying satisfied customers. The precision slightly decreased from 88.8% to 87.02%, indicating a small increase in false positives, where the model mistakenly labels some unsatisfied customers as satisfied. The specificity decreased from 36.36% to 22.73%, meaning the model is now less effective at correctly identifying negative instances, or unsatisfied customers. Lastly, the F1 score improved slightly from 91.36% to 91.57%, reflecting a slight increase in the balance between precision and recall. Based on these results, the improved model performs better in terms of recall and F1 score, but there is a trade-off with specificity. This suggests that while the improved model is better at detecting positive instances, it has become slightly more prone to misclassifying negatives as positives.

# Discussion and Next Steps

## Summary of Key Takeaways:

The project aims to predict customer satisfaction with AI tools, assisting businesses in identifying customer segments most likely to have positive experiences with such technology. Initial exploration and modeling provided valuable insights into the relationships between demographic features, AI usage, and satisfaction levels. Our initial neural network model achieved high accuracy (85%) but faced overfitting challenges, as evident from declining validation accuracy and increasing validation loss. Improvements such as introducing PCA, dropout layers, and early stopping mitigated overfitting, leading to more balanced results, including improved recall and F1 scores. While specificity decreased slightly, the improved model aligns with the project’s goal of accurately identifying satisfied customers. These findings support the hypothesis that satisfaction with AI varies across demographics and tools, validating the decision to use neural networks for capturing complex relationships.

## Further Tuning Plan:

Future changes will focus on additional model improvements and validation strategies. We plan to potentially experiment with some other methods, such as Random Forest and Gradient Boosting, to compare performance with the neural network model. Additionally, hyperparameter tuning will extend to optimizing dropout rates, patience levels for early stopping, and batch sizes to further reduce overfitting while maintaining accuracy. The analysis plan will incorporate feature selection to address potential multicollinearity among features, ensuring the model only uses variables that provide unique, non-redundant information. Cross-validation techniques will be expanded to evaluate performance consistency across different subsets of data. By refining our approach, we aim to enhance the model’s precision and specificity, creating a robust tool for predicting customer satisfaction with AI tools.

# GitHub Link

<https://github.com/Canfieldr/DSE6311-Graduate-DS-Capstone>

# 

# References

Bergen, Mark, and Lynn Doan. “Tech Giants Are Set to Spend $200 Billion This Year Chasing AI.” *Bloomberg.com*, Bloomberg, Nov. 2024, www.bloomberg.com/news/articles/2024-11-01/tech-giants-are-set-to-spend-200-billion-this-year-chasing-ai. Accessed 3 Nov. 2024.

Chollet, F., Kalinowski, T., and Allaire, J. J. (2022). Deep Learning with R. (Second Edition). Manning.

Geist, K.S. (2019) *calcSplitRatio-3*, *GitHub*. Merrimack College. Available at: https://github.com/ksgeist (Accessed: 14 November 2024).

James G., Witten D., Hastie T., & Tibshirani R. (2015). An Introduction to Statistical Learning with Applications in R (2nd ed.). Springer.

Kannan, Rathimala; Ramakrishnan, Kannan; Ersoy, Ayse Begum; Contu, Davide (2023). Customer Satisfaction Response to Artificial Intelligence Tools Usage During Online Shopping. figshare. Dataset. https://doi.org/10.6084/m9.figshare.24633105.v1

Kasaraneni, Ramana Kumar (2021). AI-Enhanced Supply Chain Collaboration Platforms for Retail: Improving Coordination and Reducing Costs. *Journal of Bioinformatics and Artificial Intelligence*, 1(1), pp. 410–450. Available at: https://biotechjournal.org/index.php/jbai/article/view/98 (Accessed: 3 November 2024).

What is the difference between one-hot and dummy encoding? (n.d.) *Data Science Stack Exchange*. Available at: https://datascience.stackexchange.com/ questions/98172/what-is-the-difference-between-one-hot-and-dummy-encoding.

# Appendix

Data Dictionary:

|  |  |  |
| --- | --- | --- |
| Variable | Data Type | Description |
| AI\_Satisfaction | String | Overall satisfaction level |
| Country | String | Country of the respondent |
| Online\_Consumer | String | Whether the consumer shops online (YES/NO) |
| Age | String | Age group of the respondent (Gen X, Gen Z, etc.) |
| Annual\_Salary | String | Salary range of the respondent (Low, Medium, High) |
| Gender | String | Gender of the respondent (Male/Female) |
| Education | String | Highest education level achieved by the respondent (Masters' Degree, University Graduate, etc.) |
| Payment\_Method\_Credit/Debit | String | Whether the respondent uses credit or debit cards for payments (YES/NO) |
| Living\_Region | String | Type of living area (Metropolitan, Rural Areas) |
| Online\_Service\_Preference | String | Preference for using online services (YES/NO) |
| AI\_Enhance\_Experience | String | Whether the respondent feels AI enhances their shopping experience (YES/NO) |
| AI\_Trust | String | (AI\_Privacy\_No\_Trust + 1 + AI\_Endorsement+1)/2 (Will create High (2), Medium (1), Low Trust(0)) |
| AI\_Usage | String | AI\_Tools variables/3 (Will create High (3), Medium (2), Low Usage(1)) |
| AI\_Tools\_Used\_Chatbots | String | Use of chatbots for assistance (YES/NO) |
| AI\_Tools\_Used\_Virtual\_Assistant | String | Use of virtual assistants (YES/NO) |
| AI\_Tools\_Used\_Voice&Photo\_Search | String | Use of voice and photo search tools (YES/NO) |
| Paymeny\_Method\_COD | String | Use of Cash on Delivery as a payment method (YES/NO) |
| Payment\_Method\_Ewallet | String | Use of e-wallets as a payment method (YES/NO) |
| Product\_Category\_Appliances | String | Interest or purchase in appliances category (YES/NO) |
| Product\_Category\_Electronics | String | Interest or purchase in electronics category (YES/NO) |
| Product\_Category\_Groceries | String | Interest or purchase in groceries category (YES/NO) |
| Product\_Category\_Personal\_Care | String | Interest or purchase in personal care category (YES/NO) |
| Product\_Category\_Clothing | String | Interest or purchase in clothing category (YES/NO) |

***Appendix***

*# Import necessary libraries*

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** random

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.decomposition **import** PCA

**from** sklearn.metrics **import** confusion\_matrix, roc\_curve, auc, ConfusionMatrixDisplay

**from** sklearn.metrics **import** accuracy\_score, recall\_score, precision\_score

**import** tensorflow **as** tf

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** Dense, Dropout

**from** tensorflow.keras.callbacks **import** EarlyStopping

**import** warnings

warnings**.**filterwarnings("ignore")

In [47]:

*# Load in the full dataset*

df **=** pd**.**read\_csv("FE\_final\_data.csv")

df**.**head()

In [48]:

*# Split the dataset into training and testing sets (78% - 22% split) based off of our calculations*

X **=** df**.**drop(columns **=** ['AI\_Satisfaction'])

y **=** df['AI\_Satisfaction']

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size **=** 0.22, random\_state **=** 3)

*# Scaling the features*

scaler **=** StandardScaler()

X\_train\_scaled **=** scaler**.**fit\_transform(X\_train)

X\_test\_scaled **=** scaler**.**transform(X\_test)

In [104]:

*# This is for producing the same results*

random**.**seed(3)

*# Defining the model*

model **=** Sequential()

model**.**add(Dense(256, activation **=** 'relu', input\_dim **=** X\_train\_scaled**.**shape[1]))

model**.**add(Dropout(0.2))

model**.**add(Dense(124, activation **=** 'relu'))

model**.**add(Dropout(0.2))

model**.**add(Dense(64, activation **=** 'relu'))

model**.**add(Dropout(0.2))

model**.**add(Dense(12, activation **=** 'relu'))

model**.**add(Dense(1, activation **=** 'sigmoid'))

*# Compiling the model*

model**.**compile(optimizer **=** 'RMSprop', loss **=** 'binary\_crossentropy', metrics **=** ['accuracy'])

*# Fitting the model*

history **=** model**.**fit(X\_train\_scaled, y\_train, epochs **=** 100, batch\_size **=** 512, validation\_split **=** 0.33)

In [105]:

*# Plotting the training history*

plt**.**plot(history**.**history['accuracy'], label **=** 'accuracy')

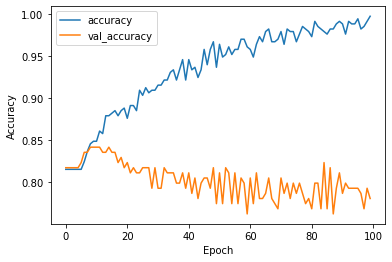
plt**.**plot(history**.**history['val\_accuracy'], label **=** 'val\_accuracy')

plt**.**xlabel('Epoch')

plt**.**ylabel('Accuracy')

plt**.**legend()

plt**.**show()



In [106]:

*# Predicting on the test set*

predictions **=** model**.**predict(X\_test\_scaled)

predicted\_class **=** (predictions **>=** 0.5)**.**astype(int)

*# Displaying the Confusion Matrix*

cf\_matrix **=** confusion\_matrix(y\_test, predicted\_class)

cmd **=** ConfusionMatrixDisplay(cf\_matrix, display\_labels **=** ['No', 'Yes'])

cmd**.**plot(cmap **=** 'mako')

cmd**.**ax\_**.**set(xlabel **=** 'Predicted', ylabel **=** 'True', title **=** 'NN: Was the User Satisfied with AI?')

*# Printing results*

print("When predicting on the test dataset, the accuracy of NN was:", np**.**round((accuracy\_score(y\_test, predicted\_class) **\*** 100), 2), "%")

print("When predicting on the test dataset, the recall of NN was:", np**.**round((recall\_score(y\_test, predicted\_class) **\*** 100), 2), "%")

print("When predicting on the test dataset, the precision of NN was:", np**.**round((precision\_score(y\_test, predicted\_class) **\*** 100), 2), "%")

print("When predicting on the test dataset, the F1 of NN was:", np.round((f1\_score(y\_test, predicted\_class) \* 100), 2), "%")

print("When predicting on the test dataset, the specificity of NN was:", np.round((recall\_score(y\_test, predicted\_class, pos\_label = 0) \* 100), 2), "%")

**5/5** ━━━━━━━━━━━━━━━━━━━━ **0s** 1ms/step

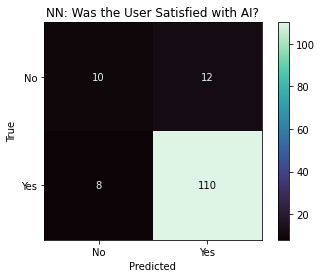
When predicting on the test dataset, the accuracy of NN was: 85.0 %

When predicting on the test dataset, the recall of NN was: 94.07 %

When predicting on the test dataset, the precision of NN was: 88.8 %

When predicting on the test dataset, the F1 of NN was: 91.36 %

When predicting on the test dataset, the specificity of NN was: 36.36 %



In [107]:

*# This is to try and stop the NN from overfitting*

*# Performing PCA on the training features.*

pca **=** PCA(n\_components **=** 0.95)

X\_train\_pca **=** pca**.**fit\_transform(X\_train\_scaled)

X\_test\_pca **=** pca**.**transform(X\_test\_scaled)

*# Creating a new model for PCA-reduced data and adding a early dropout to see if the model's accuracy will increase.*

model2 **=** Sequential()

model2**.**add(Dense(256, activation **=** 'relu', input\_dim **=** X\_train\_pca**.**shape[1]))

model2**.**add(Dropout(0.2))

model2**.**add(Dense(124, activation **=** 'relu'))

model2**.**add(Dropout(0.2))

model2**.**add(Dense(64, activation **=** 'relu'))

model2**.**add(Dropout(0.2))

model2**.**add(Dense(12, activation **=** 'relu'))

model2**.**add(Dropout(0.2))

model2**.**add(Dense(1, activation **=** 'sigmoid'))

In [108]:

*# Compile the second model*

model2**.**compile(optimizer **=** 'rmsprop', loss **=** 'binary\_crossentropy', metrics **=** ['accuracy'])

*# Define early stopping*

early\_stopping **=** EarlyStopping(monitor **=** 'val\_loss', patience **=** 10, restore\_best\_weights **=** **True**)

*# Fit the second model with early stopping*

history\_pca **=** model2**.**fit(X\_train\_pca, y\_train, epochs **=** 100, batch\_size **=** 512,

validation\_split **=** 0.33, callbacks **=** [early\_stopping])

In [109]:

*# Plot training history for PCA model*

plt**.**plot(history\_pca**.**history['accuracy'], label**=**'accuracy')

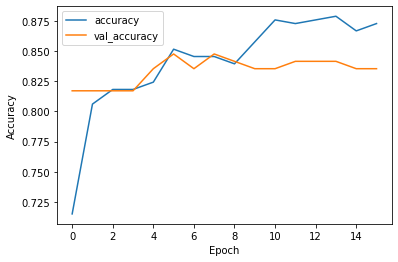
plt**.**plot(history\_pca**.**history['val\_accuracy'], label**=**'val\_accuracy')

plt**.**xlabel('Epoch')

plt**.**ylabel('Accuracy')

plt**.**legend()

plt**.**show()



In [110]:

*# Predict on the test set using PCA-reduced data*

predictions\_pca **=** model2**.**predict(X\_test\_pca)

predicted\_class\_pca **=** (predictions\_pca **>=** 0.5)**.**astype(int)

*# Displaying the Confusion Matrix*

cf\_matrix **=** confusion\_matrix(y\_test, predicted\_class\_pca)

cmd **=** ConfusionMatrixDisplay(cf\_matrix, display\_labels **=** ['No', 'Yes'])

cmd**.**plot(cmap **=** 'inferno')

cmd**.**ax\_**.**set(xlabel **=** 'Predicted', ylabel **=** 'True', title **=** 'NN: Was the User Satisfied with AI?')

*# Printing results*

print("When predicting on the test dataset, the accuracy of NN was:", np**.**round((accuracy\_score(y\_test, predicted\_class\_pca) **\*** 100), 2), "%")

print("When predicting on the test dataset, the recall of NN was:", np**.**round((recall\_score(y\_test, predicted\_class\_pca) **\*** 100), 2), "%")

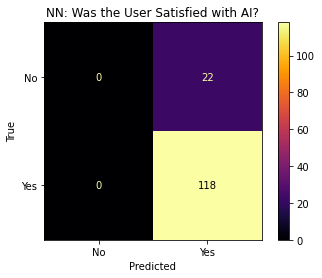
print("When predicting on the test dataset, the precision of NN was:", np**.**round((precision\_score(y\_test, predicted\_class\_pca) **\*** 100), 2), "%")

**5/5** ━━━━━━━━━━━━━━━━━━━━ **0s** 8ms/step

When predicting on the test dataset, the accuracy of NN was: 84.29 %

When predicting on the test dataset, the recall of NN was: 100.0 %

When predicting on the test dataset, the precision of NN was: 84.29 %



In [111]:

*# ROC Curve and AUC*

fpr, tpr, \_ **=** roc\_curve(y\_test, predictions)

roc\_auc **=** auc(fpr, tpr)

fpr\_pca, tpr\_pca, \_ **=** roc\_curve(y\_test, predictions\_pca)

roc\_auc\_pca **=** auc(fpr\_pca, tpr\_pca)

*# Plot ROC curves*

plt**.**figure()

plt**.**plot(fpr, tpr, color **=** 'blue', lw **=** 2, label **=** f'Model 1 (AUC = {roc\_auc:.2f})')

plt**.**plot(fpr\_pca, tpr\_pca, color**=**'red', lw **=** 2, label **=** f'Model 2 (AUC = {roc\_auc\_pca:.2f})')

plt**.**plot([0, 1], [0, 1], color **=** 'gray', lw **=** 2, linestyle **=** '--')

plt**.**xlim([0.0, 1.0])

plt**.**ylim([0.0, 1.05])

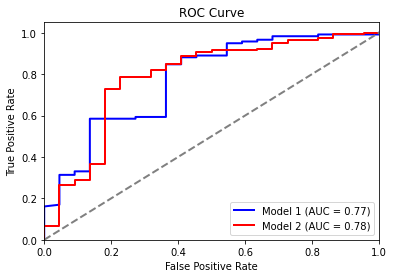
plt**.**xlabel('False Positive Rate')

plt**.**ylabel('True Positive Rate')

plt**.**title('ROC Curve')

plt**.**legend(loc **=** 'lower right')

plt**.**show()



In [ ]: