



PREDICT ONLINE PURCHASE INTENT – CUSTOMER BEHAVIOUR MODELING

MACHINE LEARNING PROJECT

SAMATRIX CONSULTING PVT LTD

GURUGRAM

contact@samatrix.io




Table of Contents

<i>Predicting Online Purchase Intent: A KNN-Based Approach to Customer Behaviour Modelling in E-Commerce</i>	2
Project Goal:.....	2
Learning Objectives	2
What to Submit:	3
Step 1 : Importing the Libraries.....	3
Step 2: Load and Preview Your Dataset	4
Step 3: Convert Text and Boolean Columns into Numbers	5
Step 4: Splitting the Data for Training and Testing	6
Step 5: Scale the Input Data to Prepare for Modeling	7
Step 6: Train a Logistic Regression Model and Make Predictions.....	8
Step 7: Add Random Noise to Your Data and Test Curse of Dimensionality.....	9
Step 8: How KNN Performance Changes with Different Values of k.....	11
Step 9: Use Cross-Validation to Check Model Performance More Reliably	12
Step 10: Use PCA to Reduce the Number of Features and Train a KNN Model.....	13
Step 11: Compare Multiple Machine Learning Models to Find the Best One	15
Step 12: Handle Class Imbalance Using SMOTE and Train a KNN Model	16
Step 13: Summarize and Present the Results of All Your Models.....	18

Predicting Online Purchase Intent: A KNN-Based Approach to Customer Behaviour Modelling in E-Commerce

Project Goal:

The goal of this project is to predict whether a user will complete a purchase during an online shopping session, based on their browsing behavior, traffic sources, time spent on product pages, and session characteristics.

Using the Online Shoppers Purchasing Intention dataset, this project applies the K-Nearest Neighbors (KNN) algorithm along with techniques such as:

- Feature scaling and dimensionality reduction (PCA)
- Class balancing using SMOTE
- Model comparison with Logistic Regression, Decision Tree, and Random Forest
- Bias-variance tradeoff analysis
- Curse of dimensionality illustration

Through this project, we aim to help e-commerce platforms:

- Understand customer engagement patterns
- Improve personalized targeting and recommendations
- Increase conversion rates by identifying high-purchase-intent sessions

This real-world application of machine learning highlights how non-parametric models like KNN can support data-driven marketing and operational decisions in digital commerce.

Learning Objectives

Data Understanding & Preprocessing

1. Load and explore real-world online retail data related to user session behavior and purchase outcomes.
2. Identify and encode categorical variables (e.g., month, visitor type) for compatibility with distance-based models like KNN.
3. Scale numerical features using standardization to prepare data for optimal KNN performance.
4. Split data into training and testing sets using stratified sampling to preserve class balance.

Model Development & Core Concepts

5. Train a K-Nearest Neighbors (KNN) model and tune the number of neighbors (K) using cross-validation.
6. Evaluate the performance of KNN using accuracy, ROC AUC, confusion matrix, and classification report.
7. Analyze the impact of noise features by adding irrelevant variables to demonstrate the curse of dimensionality.
8. Plot and interpret the bias-variance trade-off by comparing training and testing accuracy across different K values.

Model Selection & Comparison

9. Train alternative classification models including Logistic Regression, Decision Tree, and Random Forest.
10. Compare parametric and non-parametric models in terms of interpretability, flexibility, and predictive power.
11. Demonstrate the “No Free Lunch Theorem” by showing that no model universally outperforms others across all scenarios.

Dimensionality & Feature Engineering

12. Apply Principal Component Analysis (PCA) to reduce feature dimensionality and observe its effect on KNN performance.
13. Discuss trade-offs between feature richness and model simplicity in high-dimensional spaces.

Class Imbalance Handling

14. Implement SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset and improve classification of rare events (purchases).
15. Evaluate the impact of class balancing on model accuracy and sensitivity (recall for the positive class).

Visualization & Communication

16. Plot the ROC curve and visualize decision trade-offs using TPR vs FPR.
17. Summarize model findings into actionable business insights for e-commerce optimization (e.g., retargeting, personalization).
18. Reflect on ethical considerations when predicting customer behavior using behavioral data.

Dataset

https://samatrix-data.s3.ap-south-1.amazonaws.com/ML/online_shoppers_intention.csv

What to Submit:

Submission Type: Individual

Each student must submit the following:

- 1. Jupyter Notebook (.ipynb file) or python filr (.py file)**
 - a. Filename: YourFullName_online_purchase.ipynb (e.g., AnanyaKumar_online_purchase.ipynb)
 - b. Your notebook must follow the steps and structure discussed in class following the instructions in the submission guideline
- 2. Word or PDF file**
 - a. Answers “Questions for Report” in separate file

Step 1 : Importing the Libraries

1: Before you start working on the project, you need to load all the tools (libraries) that will help you handle data, build machine learning models, and evaluate results.

These libraries will allow you to:

- Work with data tables
- Perform mathematical calculations
- Visualize data using charts
- Split your data into training and testing parts
- Build models like logistic regression, decision trees, and random forests
- Measure how well your models are performing
- Handle imbalanced data by balancing the number of examples in each category

Step 2: Load and Preview Your Dataset

Now that you have imported your tools and saved the dataset file in the correct location, it's time to bring the data into your project and take your first look at it.

What to do in this step:

1. Load the Dataset File
 - Use a command to open the dataset file named `online_shoppers_intention.csv`.
 - This file contains information about user behavior on an online shopping website.
 - Make sure this file is saved in the same folder as your project notebook or script.
2. Store the Data in a Variable
 - Store the opened data in a named object (like a digital table) so that you can work with it easily in later steps.
3. Preview the First Few Rows
 - Use a command to display the first 5 rows of the dataset.
 - This gives you a quick overview of what the data looks like, including:
 - Column names
 - Types of values
 - Sample records
 - This is a good way to check if your data has loaded correctly and to start understanding what kind of information you're working with.

Questions for Report

Q1. What does the VisitorType column tell us about the users? What value do you see most often?

(Hint: Does it say whether they've visited before or if they're new?)

Q2. How many columns (features) can you see in the first five rows of the dataset? Name any three and describe what kind of information they might represent.

(Hint: Look at what each column name suggests – is it about time, user activity, or purchases?)

Q3. Based on the first few rows, does the data look clean and complete, or do you notice anything unusual or missing?

(Hint: Look for blank spaces, odd values, or repeated data.)

Q4. Choose one numeric column (like BounceRates or ExitRates). What kind of values does it show? What might a higher or lower value mean in terms of user interest?

(Hint: Think about what happens when someone quickly leaves a website.)

Q5. The Month column shows the month when the visit happened. Do you think the month could affect whether a user makes a purchase? Why or why not?

(Hint: Think about holidays, sales seasons, or shopping behavior in different months.)

Step 3: Convert Text and Boolean Columns into Numbers

Before training a machine learning model, you must make sure your data is in a format the computer can understand. Right now, some columns in your dataset contain text values (like "Feb" or "Returning_Visitor") or True/False values. These are easy for humans to read, but not for machines.

In this step, you will convert these text and boolean values into numbers so that the machine learning model can read them and learn patterns from them.

1. Convert the Month Column
 - The "Month" column contains names like "Jan", "Feb", etc.
 - These are converted into numbers like 0, 1, 2, etc.
 - This helps the model treat months as understandable values.
2. Convert the VisitorType Column
 - This column contains types of visitors like "Returning_Visitor" and "New_Visitor".
 - These are turned into numbers like 0, 1, and 2.
 - This helps the model learn whether visitor type influences buying behavior.
3. Convert Weekend Column to Numbers
 - The "Weekend" column contains True or False.
 - You will convert it into 1 (for True) and 0 (for False).
 - This helps the model know whether the session happened on a weekend or not.
4. Convert Revenue Column to Numbers
 - The "Revenue" column also contains True or False.
 - You will convert it into 1 (for a user who made a purchase) and 0 (for a user who did not).
 - This column is the one the model will try to predict, so it must be in number form.

This step is part of data preprocessing, which is necessary to build a working and accurate model.

Questions for Report

Q1. Why is it important to convert text columns like "Month" and "VisitorType" into numbers before training a machine learning model?

(Hint: Think about how machines understand data and what format they need.)

Q2. What are some examples of text values that were converted into numbers in your dataset? List at least one and mention what it was changed to.

(Hint: For example, "Feb" might be converted to 1, or "Returning_Visitor" to 0.)

Q3. How did you convert the "Weekend" column from True/False into numbers? What do the new values mean?

(Hint: Think about how 1 and 0 can be used to represent yes/no or true/false.)

Q4. The "Revenue" column is your target — the outcome you want to predict. Why does it need to be in number form, and what does each value (1 or 0) represent?

(Hint: Think about what you're teaching the model to predict — a purchase or not.)

Q5. After converting all necessary columns, how do you think this step helps prepare the data for building a successful machine learning model?

(Hint: Imagine you're explaining to someone why clean, numeric data is easier to work with.)

Step 4: Splitting the Data for Training and Testing

Now that your dataset is cleaned and ready, the next important step is to prepare it for machine learning. This step involves two things:

1. Separating the features (inputs) from the target (output you want to predict)
2. Splitting the data into two parts: one for training the model and one for testing how well it performs

What to do in this step:

1. Separate the input features and the output
 - The dataset contains many columns, but the one we are trying to predict is called Revenue. This tells us whether or not the customer made a purchase.
 - All the other columns (like number of pages visited, bounce rate, etc.) are the inputs that help the model make predictions.
 - You will create:
 - A group of columns (features) to use as input
 - A single column (Revenue) to use as the output or target
2. Split the data into training and testing sets
 - You will divide the data into two parts:
 - Training set (80%): This part is used to teach the model how the input features relate to the output.
 - Testing set (20%): This part is used to check how well the model performs on new data it hasn't seen before.
 - This helps you test the model's performance fairly and avoids overfitting (when a model memorizes the training data instead of learning from it).

- You will also make sure both training and testing sets have a similar balance of purchase and non-purchase outcomes. This is important so the model doesn't become biased toward one type of result.

Questions for Report

Q1. What is the difference between the input features and the output (target) in your dataset? Which column did you use as the target and why?

(Hint: Think about what you are trying to predict and what information helps make that prediction.)

Q2. Why do we split the dataset into training and testing sets instead of using the whole dataset for training?

(Hint: Think about how you prepare for a test by practicing first and then testing yourself with new questions.)

Q3. What percentage of the data did you use for training, and what percentage for testing? Why is it important to have both?

(Hint: A balanced split helps the model learn well and also prove its performance.)

Q4. What does it mean to 'stratify' the split when dividing the data? Why might this be helpful in your project?

(Hint: Think about keeping the same balance of outcomes in both sets so the model doesn't get biased.)

Q5. If your model performs well on the training set but poorly on the testing set, what might that tell you?

(Hint: Consider what it means when a model learns the training data too well but struggles with new data.)

Step 5: Scale the Input Data to Prepare for Modeling

Now that you have split your data into training and testing sets, the next important step is to scale the input features. Many machine learning models work better when all input values are on the same scale — in other words, when the numbers are not too large or too small compared to each other.

This is especially important when your dataset contains columns with very different ranges. For example, "Age" might range from 18 to 80, while "Bounce Rate" could be between 0.00 and 1.00. If you don't adjust these, the model might think "Age" is more important just because it has bigger numbers — even if it's not.

What to do in this step:

1. Use a scaling tool
 - You will use a method called "standard scaling."
 - This method will adjust each column so that:
 - The average value becomes 0

- The spread of the values becomes consistent
 - This helps the model focus on patterns, not on the size of the numbers.
- 2. Fit the scaler using only the training data
 - First, calculate the scaling values (like the average and standard deviation) based only on the training data.
 - This is important because we want the model to learn only from the training set, not from the testing set.
- 3. Apply the same scaling to the test data
 - Use the scaling values from the training data to adjust the test data in the same way.
 - This makes sure the test data is on the same scale without introducing any unfair advantage.

Why this step is important:

- It makes the model's learning process fairer and more accurate.
- It prevents features with large numbers from dominating the model's decisions.
- It improves the performance and speed of many models, especially those that rely on distance calculations or gradients.
- It ensures your training and testing data are prepared in a consistent way.

Questions for Report

Q1. Why is it important to scale the input features before training a machine learning model?

(Hint: Think about what happens when some columns have very large numbers and others have very small ones.)

Q2. What does it mean to 'standardize' or 'scale' a feature? What changes in the data after this process?

(Hint: Consider what happens to the average and spread of the values.)

Q3. Why do we use only the training data to fit the scaler and not the test data?

(Hint: Think about how the model should only learn from training data to avoid unfair influence.)

Q4. What could happen if you forget to scale your features in a project where feature values vary a lot?

(Hint: Would some features unfairly affect the model's decisions?)

Q5. After scaling, did you notice any visible changes in your input data values? How are they different from the original values?

(Hint: Think about whether the values became smaller or more centered around zero.)

Step 6: Train a Logistic Regression Model and Make Predictions

Now that your data is ready (scaled, split, and cleaned), it's time to build and train your first machine learning model. In this step, you will use a model called Logistic Regression to help predict whether a user is likely to make a purchase or not.

What to do in this step:

1. Create the Logistic Regression model
 - You will first create the model using a method called logistic regression.
 - This model is perfect for predicting things that have two possible outcomes, like:
 - Yes or No
 - Purchase or No Purchase
 - True or False
2. Train the model using your training data
 - Use the scaled training features and the correct answers to teach the model.
 - This is like giving the model many examples to help it learn the connection between a user's behavior and whether they made a purchase.
3. Use the model to make predictions
 - After training, the model will be ready to make predictions.
 - You will now give it the test data (new examples it hasn't seen before).
 - The model will use what it has learned to predict whether those users are likely to make a purchase.

Questions for Report

Q1. What kind of problem is Logistic Regression designed to solve? Why is it a good choice for this project?

(Hint: Think about whether you're predicting a category like Yes/No or a number.)

Q2. What does the model learn during the training process? Where does it get the information from?

(Hint: Consider how the model connects the input features to the actual results.)

Q3. After training the model, what kind of predictions did it make on the test data? What do the predicted values represent?

(Hint: Are the predictions about users who will make a purchase or not?)

Q4. Why is it important to test the model on new data (testing set) instead of the same data it was trained on?

(Hint: Think about how we check whether the model can work with unseen examples.)

Q5. If your model predicts that a user is likely to make a purchase, what real-life decision might a business take based on that?

(Hint: Could the business offer a discount, send a reminder email, or personalize recommendations?)

Step 7: Add Random Noise to Your Data and Test Curse of Dimensionality

In this step, you will explore an important concept in machine learning: how extra, irrelevant data can affect your model's performance.

Imagine you're trying to make a decision based on useful facts — like how many pages a user visited or how long they stayed on the site. Now, imagine someone gives you 10 extra pieces of completely random information that have nothing to do with the outcome. Will you still be able to make a good decision? That's exactly what we are testing here.

What to do in this step:

1. Add 10 columns of random values to your training and test data
 - You will create fake data — just random numbers between 0 and 1.
 - These numbers have no real meaning and are not related to whether a user makes a purchase.
 - These new random columns are called noise.
2. Attach the noise to your existing training and test sets
 - This means adding the 10 random columns to the end of your real data.
 - Your new datasets will contain both real features and fake, useless ones.
3. Train a new model using K-Nearest Neighbors (KNN)
 - You will train a KNN model using the noisy training data.
 - KNN makes predictions by comparing each user to others who are most similar.
4. Use the model to make predictions on noisy test data
 - You will test the model using the test set that also has noise added to it.
 - Then, you will calculate the model's accuracy to see how well it performs.

Questions for Report

Q1. What kind of data did you add to the original dataset in this step, and why was it called 'noise'?

(Hint: Think about whether the new columns contained meaningful or random information.)

Q2. After adding noise, how many new columns were included in your training and test datasets? What do these columns represent?

(Hint: Were they real features or just random numbers?)

Q3. Which machine learning model did you use for this step, and how does it make predictions?

(Hint: Think about how K-Nearest Neighbors works by comparing similar cases.)

Q4. How did the model's accuracy change after adding the noise? Did it perform better, worse, or about the same compared to before?

(Hint: Try to describe any increase or drop in accuracy.)

Q5. What did you learn about the importance of using the right features in a machine learning model?

(Hint: Consider whether irrelevant data helps or hurts the model's ability to make good predictions.)

Step 8: How KNN Performance Changes with Different Values of k

Now that you have built and tested a KNN model with one specific value of k (number of neighbors), it's time to run an experiment. In this step, you will see how changing the value of k affects the model's accuracy on both the training and test data.

This experiment helps you understand the important idea of the bias-variance tradeoff, which means finding the right balance between a model that is too simple and one that is too complex.

What to do in this step:

1. Test many different k values
 - Try values of k from 1 to 50.
 - For each value, you will build a new KNN model and record its accuracy.
2. Check the accuracy on training data and test data
 - For each value of k , measure how well the model performs on:
 - Training data (how well the model remembers what it has seen)
 - Test data (how well the model handles new, unseen cases)
 - Store these results for comparison.
3. Plot the results on a graph
 - Create a chart that shows k values on the x-axis and accuracy on the y-axis.
 - Include two lines: one for training accuracy and one for test accuracy.
 - This graph will help you visually understand how performance changes as k increases.
4. Save your graph as an image
 - This image is part of your project results and can be included in your final report or presentation.

What to look for:

- When k is too low (e.g., 1), the model might perform very well on training data but poorly on test data. This means the model is **overfitting** — it is memorizing too much and not generalizing well.
- When k is too high (e.g., 40 or 50), the model becomes too simple and might miss important patterns. This is called **underfitting**.
- Your goal is to find a **balanced value of k** where the test accuracy is high and stable.

Questions for Report

Q1. What does the value of k in the KNN model represent, and why do we experiment with different values of k ?

(Hint: Think about how the model uses neighbors to make predictions.)

Q2. How did the training accuracy and test accuracy change as you increased the value of k from 1 to 50?

(Hint: Were there any patterns you noticed in the graph, such as accuracy going up, down, or staying steady?)

Q3. At low k values (like $k = 1$ or 2), was the model more likely to memorize the training data or generalize well to new data? What is this situation called?

(Hint: Does the model overfit or underfit when it's too sensitive to training examples?)

Q4. At high k values (like $k = 40$ or 50), did the model become too simple and miss important patterns? What is this situation called?

(Hint: What happens when the model can't capture the differences in the data?)

Q5. Based on your graph, what value of k would you recommend using in the final model, and why?

(Hint: Look for the value where test accuracy is highest and most stable.)

Step 9: Use Cross-Validation to Check Model Performance More Reliably

So far, you've trained your model and tested it once on a fixed portion of the data. But what if the results change depending on which part of the data was used? To get a more reliable and stable understanding of your model's performance, you'll now use a method called cross-validation.

In this step, you will perform **10-fold cross-validation** using the K-Nearest Neighbors (KNN) model with 5 neighbors. This means the training data will be split into 10 parts, and the model will be tested 10 times — each time with a different part of the data acting as the test set.

What to do in this step:

1. Create a KNN model with 5 neighbors
 - Just like before, use the KNN model where the prediction is based on the 5 most similar past users.
 - This model will be used to test how well it performs across different parts of the training data.
2. Use 10-fold cross-validation
 - The training data is divided into 10 equal parts.
 - In each round:
 - 9 parts are used to train the model.
 - The 1 remaining part is used to test it.
 - This is repeated 10 times so that every part gets a turn to be the test set.
3. Collect and store the results
 - You will get 10 accuracy scores — one from each round.
 - These scores will help you understand the overall performance of the model, not just on one random split.

Why this step:

- Cross-validation helps you evaluate the model more fairly and thoroughly.
- Instead of relying on just one test set, you test the model on multiple subsets of the data.

- This method reduces the chances of accidental bias or lucky/unlucky test splits.
- It builds more confidence in your model's ability to work well on new data.

Example:

Imagine you're testing a student's knowledge. Instead of giving them just one final test, you give them 10 mini-tests on different topics. If they perform well consistently, you can trust they understand the subject better.

Questions for Report

Q1. What is the main purpose of using cross-validation in a machine learning project?

(Hint: Think about why testing the model multiple times is better than just once.)

Q2. In 10-fold cross-validation, how is the training data divided and used? Describe the process in your own words.

(Hint: How many parts is the data split into? What happens in each round?)

Q3. Why did we use only the training data for cross-validation and not the test data?

(Hint: Think about what would happen if we used the test data during training.)

Q4. What did you observe about the model's performance across the 10 folds? Were the accuracy scores similar or different? What does that tell you?

(Hint: Try to describe how consistent the model was.)

Q5. How does cross-validation help you feel more confident about the model's ability to work on new, unseen data?

(Hint: Think about how repeating the test with different data builds trust in the model's results.)

Step 10: Use PCA to Reduce the Number of Features and Train a KNN Model

Sometimes, datasets contain a large number of features (columns), and not all of them are equally important. Some may carry overlapping or less useful information. To deal with this, we can use a technique called PCA (Principal Component Analysis).

In this step, you will use PCA to reduce the number of features in your data to just a few important ones. Then, you'll use this simplified data to train a new KNN model and check its accuracy. This helps you learn how PCA affects both model performance and data efficiency.

What to do in this step:

1. Apply PCA to reduce your input features
 - Use PCA to transform your original input data into a smaller number of new features (called principal components).
 - These new features still carry most of the important information but in a simpler form.
 - In this project, you will reduce your data to just 5 components.

2. Fit PCA only on the training data
 - Allow PCA to learn patterns only from the training data.
 - This ensures that your model doesn't get influenced by the test data during training.
3. Transform both training and test data using PCA
 - Apply the same transformation to your test data.
 - This ensures that both training and testing sets are in the same format.
4. Train a new K-Nearest Neighbors (KNN) model
 - Use the simplified data (after PCA) to train a new KNN model.
 - Use 5 neighbors for prediction, just like in previous steps.
5. Test the model and record its accuracy
 - Use the test data (after PCA) to evaluate how well the model performs.
 - Record the accuracy score so you can compare it with other models you've built earlier.

Why this step:

- PCA helps you reduce the number of features, which can make models faster and sometimes more accurate.
- It's a powerful tool for dealing with high-dimensional data (data with many columns).
- It helps prevent overfitting by removing noise and focusing only on the most important patterns.
- Comparing this model's accuracy with previous models will show you whether PCA helped or not.

Example:

Imagine you're analyzing customer behavior using 20 survey questions. PCA helps you reduce them to just 5 key themes that explain most of the behavior. This makes it easier for the model to learn and make predictions.

Questions for Report

Q1 What is PCA (Principal Component Analysis), and why did you use it in your project?
(Hint: Think about reducing the number of columns while keeping the important information.)

Q2. How many features (principal components) did you reduce your data to using PCA? Why is it helpful to reduce the number of features?
(Hint: Consider how fewer features might help with speed or accuracy.)

Q3. After applying PCA, did you use the same transformation on both the training and test data? Why is this important?
(Hint: Think about keeping both datasets in the same format.)

Q4. How did the accuracy of your PCA-based KNN model compare with the KNN model that used the full set of features?
(Hint: Was the model more accurate, less accurate, or about the same?)

Q5. What did you learn about the trade-off between simplicity (fewer features) and model performance? Would you recommend using PCA in future projects? Why or why not? (Hint: Think about whether simpler data helped or hurt your model.)

Step 11: Compare Multiple Machine Learning Models to Find the Best One

Now that you have trained and tested a few individual models like KNN and Logistic Regression, it's time to go one step further: compare several models side by side to find out which one performs best.

In this step, you will:

- Train four different models using the same training data
- Test all of them using the same test data
- Measure how well each model performs using two important metrics: accuracy and AUC (Area Under the ROC Curve)

This helps you choose the best model based on evidence, not guesswork.

What to do in this step:

1. List the models you want to compare
 - You will test four common classification models:
 - K-Nearest Neighbors (KNN)
 - Logistic Regression
 - Decision Tree
 - Random Forest
 - Each model will be trained using the same data, so the comparison is fair.
2. Train each model using the scaled training data
 - Go through each model one by one.
 - Train the model by giving it the input features and the correct answers (whether the user made a purchase or not).
3. Test each model on the test data
 - After training, use each model to make predictions on the unseen test data.
 - This shows how well the model can generalize to new users it has never seen before.
4. Calculate two performance scores for each model
 - Accuracy: Tells you how many predictions the model got right overall.
 - AUC (Area Under the Curve): Tells you how well the model can separate positive outcomes (purchases) from negative ones.
 - These scores will be stored in a results table so you can easily compare the models.
5. Store and organize the results
 - For each model, save its name and its scores (accuracy and AUC).
 - This gives you a clear picture of which model is performing best overall.

Why this step:

- It helps you make a smart, informed decision about which model to use in your final project output.

- Different models have different strengths. Some may be faster, some more accurate, and some better at detecting rare cases.
- This comparison helps you understand the trade-offs between different types of models.
- In real-world projects, comparing models is a critical step before launching a predictive system.

Questions for Report

Q1. Why is it useful to compare multiple models instead of using just one?

(Hint: Think about how different models might behave differently on the same data.)

Q2. Which model performed the best in terms of accuracy? What does this tell you about how well it predicted overall results?

Q3. Which model had the highest AUC score? What does a high AUC score mean in the context of this project?

(Hint: Consider what it says about the model's ability to separate buyers from non-buyers.)

Q4. Were there any differences in performance between models? What do these differences suggest about the strengths or weaknesses of each model?

Q5. Based on both accuracy and AUC, which model would you choose for this project and why? Explain your reasoning clearly.

Step 12: Handle Class Imbalance Using SMOTE and Train a KNN Model

In many real-life datasets, one type of outcome (like “no purchase”) appears much more often than the other (like “purchase”). This is called class imbalance, and it can cause machine learning models to be biased — they may mostly predict the majority class and ignore the important minority cases.

To solve this, we use a technique called SMOTE (Synthetic Minority Over-sampling Technique). It creates new, artificial examples of the minority class so the model has more balanced data to learn from.

In this step, you will use SMOTE to balance the training data, then train a K-Nearest Neighbors (KNN) model on the balanced data, and finally test its performance on the test data.

What to do in this step:

1. Apply SMOTE to the training data
 - SMOTE will create new data points for the minority class.
 - These new examples are not copied; they are generated based on the patterns in your data.
 - This helps your model learn to recognize both classes more equally.
2. Train a KNN model using the resampled (balanced) data

- Use the same KNN method you've been using, but this time feed it the balanced dataset.
- The model will now be able to learn from both majority and minority examples.
- 3. Test the model using the original test data
 - Keep the test data unchanged to make sure your evaluation is fair.
 - This helps you check if the SMOTE-trained model performs better than before.
- 4. Record the accuracy
 - Measure how many predictions the model got correct on the test data.
 - Compare this accuracy to the previous models to see if balancing the data helped.

Why this step:

- It helps you deal with **imbalanced datasets**, which are very common in real-life problems (like fraud detection or disease prediction).
- It gives your model a better chance to learn from the **less frequent but important outcomes**.
- It teaches you a valuable technique (SMOTE) that is used in many data science and machine learning projects.

Real-life example:

Imagine you are predicting whether a customer will buy a product. If only 2% of customers actually buy something, a model might just always say “no” to everyone. SMOTE helps you train the model to recognize and **not ignore** that small group of buyers — which could be the most important group for a business.

Questions for Report

Q1. What is class imbalance, and why can it be a problem in machine learning models?
(Hint: Think about what happens if the model mostly sees one type of outcome.)

Q2. What does SMOTE do, and how does it help solve the class imbalance issue?
(Hint: Try to explain how SMOTE creates new examples for the less frequent class.)

Q3. Why did we apply SMOTE only on the training data and not the test data?
(Hint: Think about keeping the test data untouched for a fair evaluation.)

Q4. How did the accuracy of the KNN model trained on resampled data compare with the original KNN model? Was there any improvement?
(Hint: Look at your accuracy scores before and after using SMOTE.)

Q5. Would you recommend using SMOTE for future projects with imbalanced data? Why or why not?
(Hint: Think about the effect it had on your model's ability to detect both classes.)

Step 13: Summarize and Present the Results of All Your Models

Now that you've trained multiple machine learning models, applied advanced techniques like PCA and SMOTE, and measured different accuracy scores, it's time to bring everything together.

This final step helps you **compare all your models** side by side, so you can make a well-informed decision about which one works best. It's like preparing your final project results for a report or presentation.

What to do in this step:

1. Print the results of key experiments
 - Display the accuracy of important models such as:
 - Logistic Regression
 - KNN with added noise
 - PCA + KNN
 - KNN with SMOTE-balanced data
 - Show the average KNN accuracy using cross-validation
2. Print a comparison table of all the models you trained
 - Include Accuracy and AUC (Area Under Curve) scores for each model.
 - Models include: KNN, Logistic Regression, Decision Tree, and Random Forest.
 - This will help you understand which models performed best overall and which ones might be useful in real-world situations.
3. Highlight the saved ROC curve plot
 - Mention that your model evaluation chart (ROC curve) has been saved as an image file.
 - You can include it in your project report to visually show how well the models separate the two classes (buyers vs non-buyers).

Why this step:

- It gives you a clear, easy-to-read summary of how your models performed.
- You can now compare models based on real results, not just guesses.
- This step helps you decide which model to use in the final deployment or recommendation.
- You practice the important skill of presenting machine learning results in a professional way — a must-have for real-world data projects.

Questions for Report

Q1. Among all the models you tested, which one gave the highest accuracy? What does this tell you about the model's performance?

(Hint: Look at the summary where each model's accuracy is printed.)

Q2. Which model had the highest AUC score? Why is AUC important in classification problems like this one?

(Hint: AUC tells you how well the model separates buyers from non-buyers.)

Q3. Was there a model that had high accuracy but low AUC, or vice versa? What does that mean about the way it predicts outcomes?

Q4. Which model would you choose for this project if you had to use it in the real world? Explain your choice based on both performance and simplicity.

Q5. What are two important lessons you learned from comparing different models? How will this help you in future machine learning projects?