Analyzing and Predicting E-Commerce Using Machine Learning

Dara Alegre

ITSC-3156 Final Project

1 - Introduction

In the fast-paced world of e-commerce, identifying factors that make certain products emerge as "Best Sellers" is essential for both online retailers and market analysts. Top-selling products often exhibit recognizable patterns, including pricing strategies, discount levels, and their respective categories. By examining these attributes, data-driven techniques can provide valuable predictions about which products are most likely to succeed. This project investigates the use of two widely accessible machine learning methods such as Logistic Regression and Random Forest to estimate the likelihood of a product becoming a best seller using a dataset containing thousands of items. Beyond building predictive models, the study aims to compare their performance, interpret results, and assess which method most effectively captures the key factors influencing product success.

Github Repository: https://github.com/dara-aleg/ITCS3156-FinalProject

1.1 - Problem Statement

In the competitive landscape of e-commerce, online retailers need reliable methods to identify which products are likely to become best sellers. Despite the availability of large product datasets, predicting top-performing items remains challenging due to the variety of influencing factors, including price, discount, category, and other product attributes. This project seeks to address this challenge by applying machine learning techniques, specifically Logistic Regression and Random Forest, to estimate the probability of a product becoming a best seller. The study also aims to evaluate and compare the predictive performance of these models, while uncovering the key factors that drive product success in online marketplaces.

1.2 - Motivation and Challenges

Predicting which products will become best sellers is critical for online retailers, as accurate forecasts can optimize inventory management, pricing strategies, marketing campaigns, and product recommendations, ultimately boosting sales and customer satisfaction. Leveraging machine learning techniques allows businesses to transform large, complex product datasets into actionable insights, helping identify the key factors that drive product success and supporting more informed strategic decisions.

Despite the potential benefits, predicting best-selling products presents several challenges. Product datasets often contain numerous features with complex interactions, making it difficult for models to capture the most influential factors. Only a small fraction of items typically achieve best-seller status, creating class imbalance issues that can affect predictive accuracy. Data inconsistencies, missing values, and errors further complicate model training, and there is often a trade-off between model interpretability and predictive performance, limiting the ability to extract actionable insights.

1.3 - Summary of Approach

This project begins by conducting Exploratory Data Analysis and data preprocessing to gain a deep understanding of the e-commerce product dataset and prepare it for modeling. The analysis includes examining distributions of key features such as product price, discount percentage, product category, and other attributes that may influence sales performance. Visualizations and statistical summaries help identify patterns, outliers, and potential correlations that impact the likelihood of a product becoming a best seller.

Following the analysis, two machine learning models, Logistic Regression and Random Forest, are implemented to predict the probability that a product achieves best-seller status. Logistic Regression serves as a baseline linear model, while Random Forest captures nonlinear relationships and complex interactions between features. Both models are trained on the preprocessed dataset and evaluated using accuracy, precision, recall, F1-score, and confusion matrices. Comparing the models provides insight into which approach is most effective and helps identify the key factors that contribute to product success, offering actionable guidance for retailers and market analysts.

2 - Data/Environment

2.1 - Dataset Introduction

The dataset from Kaggle, provides detailed information on roughly 80,000 products from an e-commerce platform. It includes attributes such as product price, discount percentage, category, and other factors that could affect consumer purchasing behavior. Its large size and diversity enable an in-depth analysis across a broad range of product types and price levels. Prior cleaning has addressed missing values, formatting issues, and errors, ensuring a solid foundation for reliable analysis.

2.2 Data Visualization

Import Utils

Exploratory Data Analysis was conducted to understand the distribution and relationships among product features. Price and discount distributions were visualized to identify outliers and patterns, while categorical features such as product category were analyzed to determine their

prevalence and potential influence on sales. Correlation analysis revealed which features are most strongly associated with best-seller status, highlighting factors such as competitive pricing and popular categories. No missing values or major inconsistencies were found after cleaning, providing a reliable basis for model training.

2.2 - Data Processing

In [39]:

```
# %pip install numpy pandas matplotlib scikit-learn

from typing import List, Dict, Tuple, Callable
import os
import gc
import traceback
import warnings
from pdb import set_trace

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
```

2.2.1 Load Dataset

We use `read_csv` from `pandas` to load our data from the CSV file.

In [40]:

```
products = pd.read_csv(r'products.csv', low_memory=False)
```

Then, we can `display()` the `products` dataframe

In [41]:

```
display(products)
```

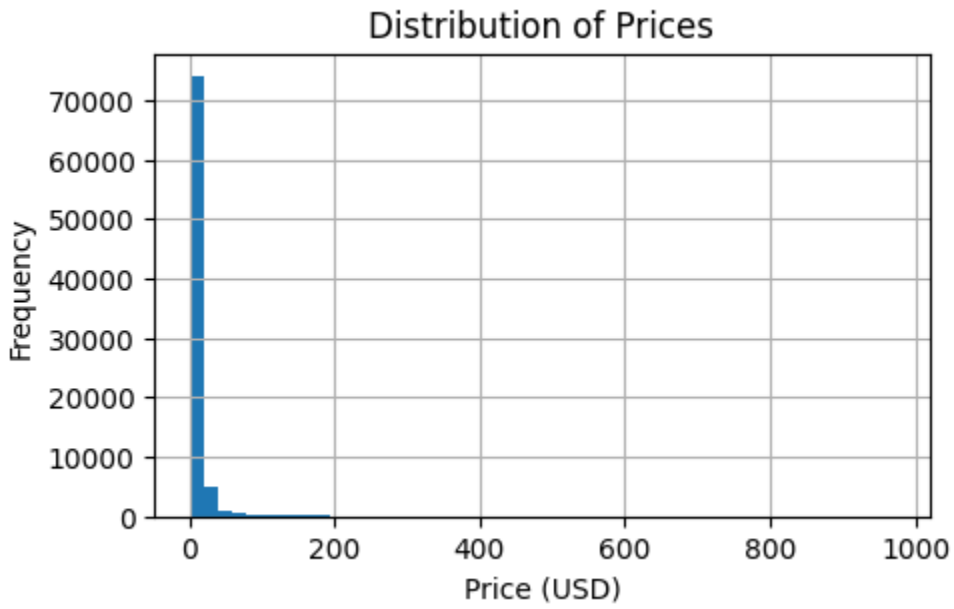
	Unnamed: 0	rank	rank_category	price_usd	pct_discount	qty_sold	product_title	category_name	color_count	img_source_url	black_friday_
0	0	#1 Best Sellers	Give Gifts	2.03	22.0	NaN	1pc Rechargeable Deep Tissue Muscle Handheld M...	appliances	NaN	NaN	
1	1	#4 Best Sellers	Top rated Portable Fans	6.48	20.0	NaN	1pc Portable Hanging Neck Fan	appliances	NaN	NaN	
2	2	NaN	NaN	1.80	NaN	400.0	1pc Pink Colored Curved Eyelash Curler False E...	appliances	NaN	NaN	
3	3	NaN	NaN	0.88	72.0	5600.0	1 Mini Portable Handheld Fan With 2 Aa Batteri...	appliances	NaN	NaN	
4	4	#6 Best Sellers	Oral Irrigators	12.06	40.0	NaN	Wit Water Flosser,Portable Oral Irrigator With...	appliances	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	

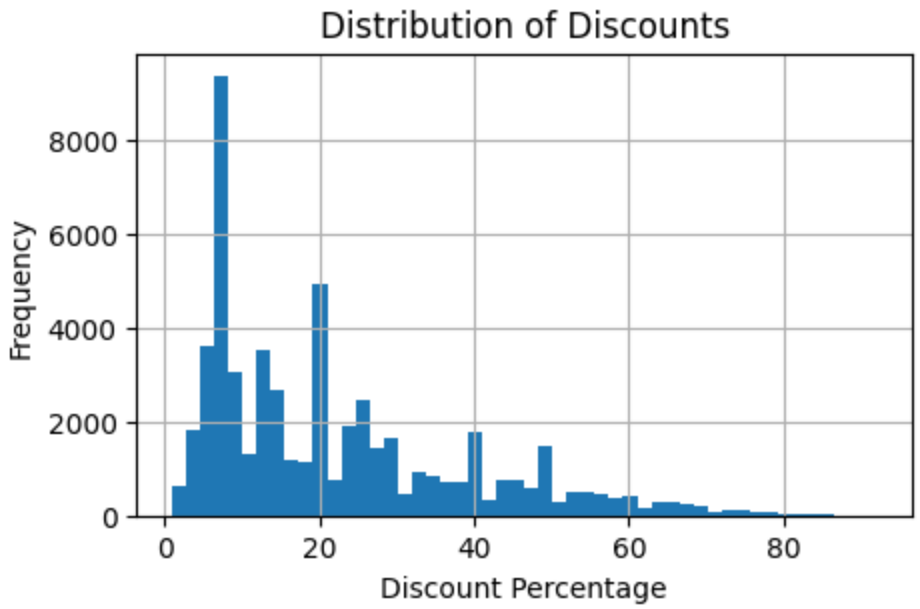
2.2.2 Visualizations

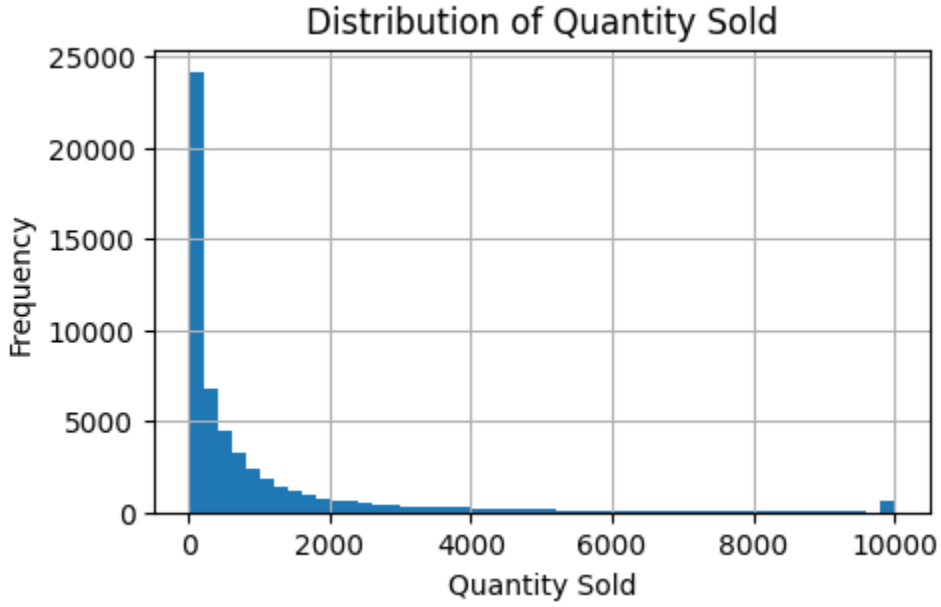
Plots of Distribution of Prices, Discounts, and Qty. Sold

In [42]:
```
# Plot the distribution of prices
plt.figure(figsize=(5, 3))
products['price_usd'] = pd.to_numeric(products['price_usd'], errors='coerce')
products['price_usd'].dropna().hist(bins=50)
plt.title('Distribution of Prices')
plt.xlabel('Price (USD)')
plt.ylabel('Frequency')
plt.show()

# Plot the distribution of discounts
plt.figure(figsize=(5, 3))
products['pct_discount'].dropna().hist(bins=50)
plt.title('Distribution of Discounts')
plt.xlabel('Discount Percentage')
plt.ylabel('Frequency')
plt.show()

# Plot the distribution of quantity sold
plt.figure(figsize=(5, 3))
products['qty_sold'].dropna().hist(bins=50)
plt.title('Distribution of Quantity Sold')
plt.xlabel('Quantity Sold')
```

```
plt.ylabel('Frequency')
plt.show()
```







List of Unique Product Categories

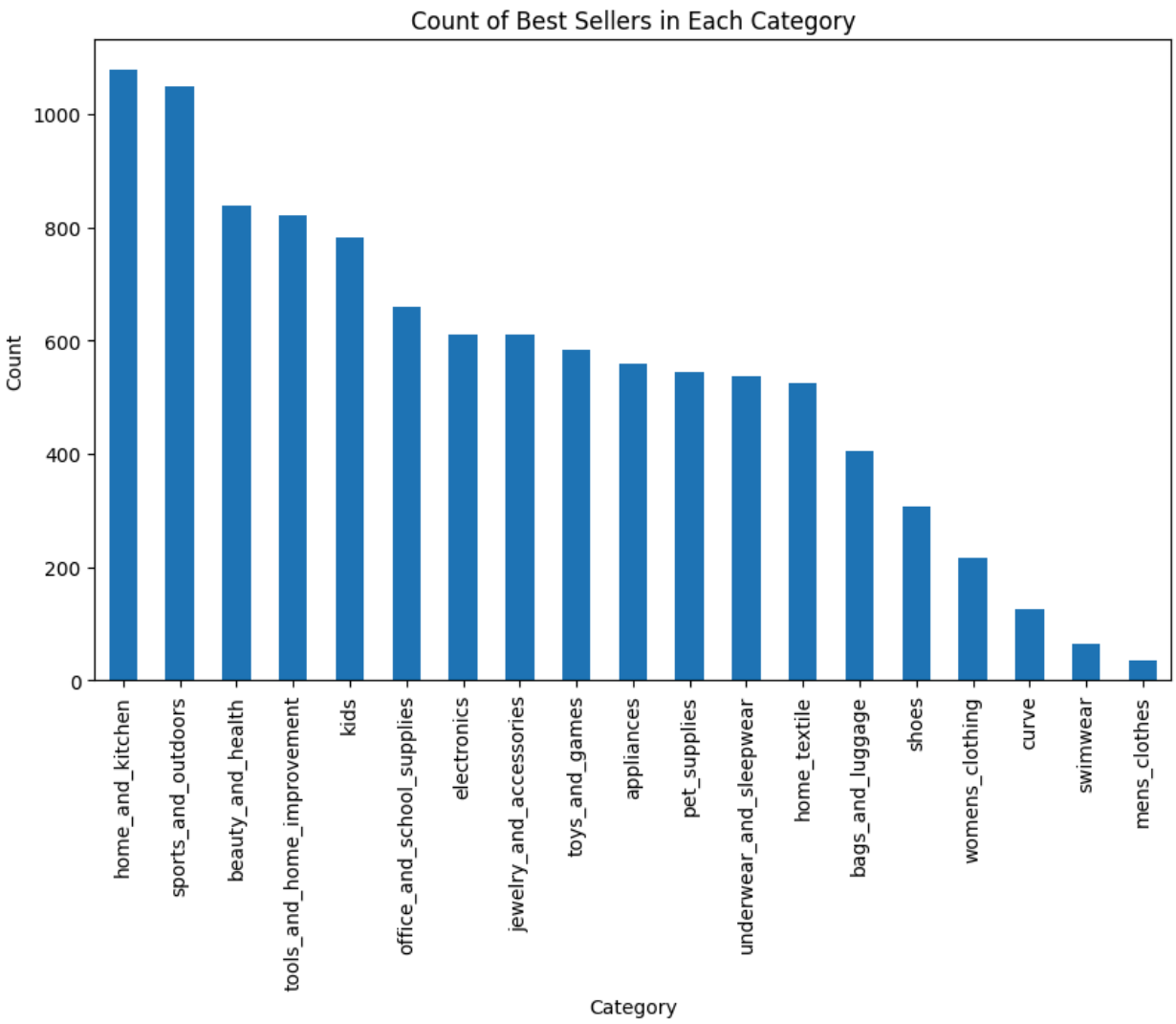
In [43]:
```
# Print all unique product categories
unique_categories = products['category_name'].unique()
print(f"Product Categories: \n {unique_categories} \n")
print(f"Number of Categories: {unique_categories.size}")
```

```
Product Categories:
 ['appliances' 'office_and_school_supplies' 'underwear_and_sleepwear'
 'mens_clothes' 'curve' 'swimwear' 'womens_clothing' 'sports_and_outdoors'
 'home_textile' 'beauty_and_health' 'home_and_kitchen'
 'jewelry_and_accessories' 'electronics' 'toys_and_games' 'shoes'
 'baby_and_maternity' 'bags_and_luggage' 'kids' 'automotive'
 'tools_and_home_improvement' 'pet_supplies']

Number of Categories: 21
```

According to the calculations there are 21 unique categories. Since we are interested in the Best Sellers, we will be plotting the counts of Best Sellers in each product category.

Plot of Counts of Best Sellers by Category

In [44]:
```
# Extract bestseller information
bestsellers = products[products['rank'].str.contains('Best Sellers', na=False)]

# Plot the count of bestsellers in each category
plt.figure(figsize=(10, 6))
bestseller_counts = bestsellers['category_name'].value_counts()
bestseller_counts.plot(kind='bar')
plt.title('Count of Best Sellers in Each Category')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
display(bestsellers)

# Print the number of bestsellers
print(f"Number of Bestsellers: {bestsellers.shape[0]}")
```


Count of Best Sellers in Each Category
Count
0
200
400
600
800
1000
home_and_kitchen
sports_and_outdoors
beauty_and_health
tools_and_home_improvement
kids
office_and_school_supplies
electronics
jewelry_and_accessories
toys_and_games
appliances
pet_supplies
underwear_and_sleepwear
home_textile
bags_and_luggage
shoes
womens_clothing
curve
swimwear
mens_clothes
Category

	Unnamed: 0	rank	rank_category	price_usd	pct_discount	qty_sold	product_title	category_name	color_count	img_source_url	black_friday_
0	0	#1 Best Sellers	Give Gifts	2.03	22.0	NaN	1pc Rechargeable Deep Tissue Muscle Handheld M...	appliances	NaN	NaN	
1	1	#4 Best Sellers	Top rated Portable Fans	6.48	20.0	NaN	1pc Portable Hanging Neck Fan	appliances	NaN	NaN	
4	4	#6 Best Sellers	Oral Irrigators	12.06	40.0	NaN	Wit Water Flosser,Portable Oral Irrigator With...	appliances	NaN	NaN	
5	5	#10 Best Sellers	Refrigerators & Freezers	2.70	NaN	NaN	1pc Ice Pop Mold, Plastic Ice Cream Mold, Froz...	appliances	NaN	NaN	
6	6	#8 Best Sellers	Other Household Appliances	3.50	NaN	NaN	Mini Pocket Bluetooth Thermal Printer, Portabl...	appliances	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	
81514	81514	#6 Best Sellers	Aquarium Appliances	3.76	20.0	NaN	Vibrant Aquarium Enhancement: Waterproof Fluo...	pet_supplies	NaN	NaN	
81592	81592	#9 Best Sellers	Pet Accessories Sets	3.36	20.0	NaN	1 Set Black Dog Paw Print Happy Birthday Party...	pet_supplies	NaN	NaN	
81601	81601	#1 Best Sellers	Small Animal Cages & Accessories	0.86	67.0	NaN	1pc/2pcs Rabbit Grass Mat For Hamsters, Lop-Ea...	pet_supplies	NaN	NaN	
81614	81614	#1 Best Sellers	Small Pet Clothg	2.52	10.0	NaN	1pc Strawberry Design Rabbit Tank For Small An...	pet_supplies	NaN	NaN	
81863	81863	#6 Best Sellers	Farm Cages & Accessories	5.04	20.0	NaN	2pcs Green Chicken Wire Netting For Flower Arr...	pet_supplies	NaN	NaN	

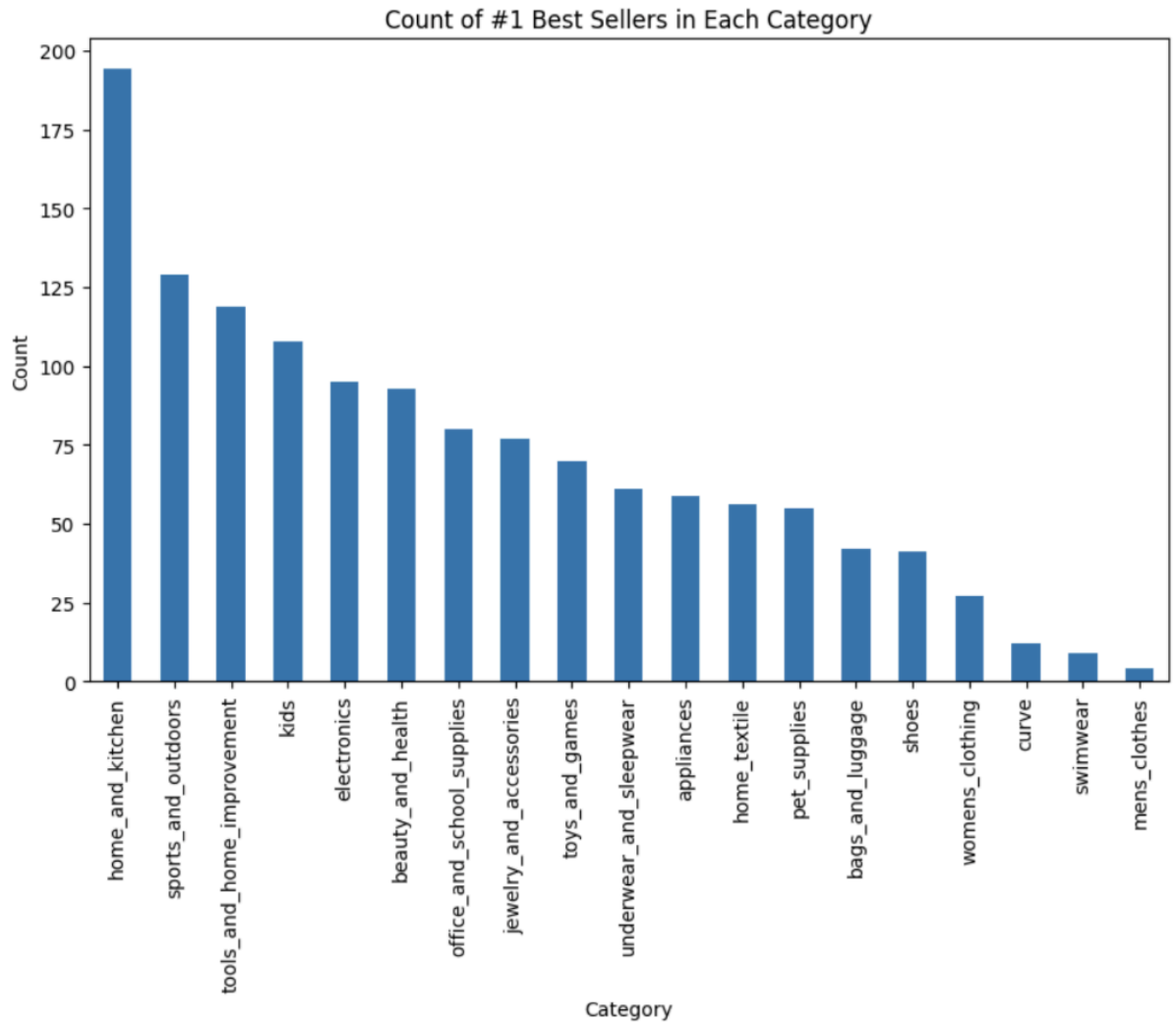
```
Number of Bestsellers: 10353
```

The most popular categories are Home and Kitchen, Beauty & Health, Sports & Outdoors, Tools & Home Improvement, each having over 800 best-sellers.

We can then look at #1 Best Sellers per category.

Plot of counts of #1 Best Sellers by Category

```
In [45]: # Filter for #1 Best Sellers
number_one_bestsellers = bestsellers[bestsellers['rank'] == '#1 Best Sellers']

# Plot the count of #1 bestsellers in each category
plt.figure(figsize=(10, 6))
number_one_bestseller_counts = number_one_bestsellers['category_name'].value_counts()
number_one_bestseller_counts.plot(kind='bar')
plt.title('Count of #1 Best Sellers in Each Category')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()

# Print the number of #1 Best Sellers
print(f"Number of #1 Best Sellers: {number_one_bestsellers.shape[0]}")
```



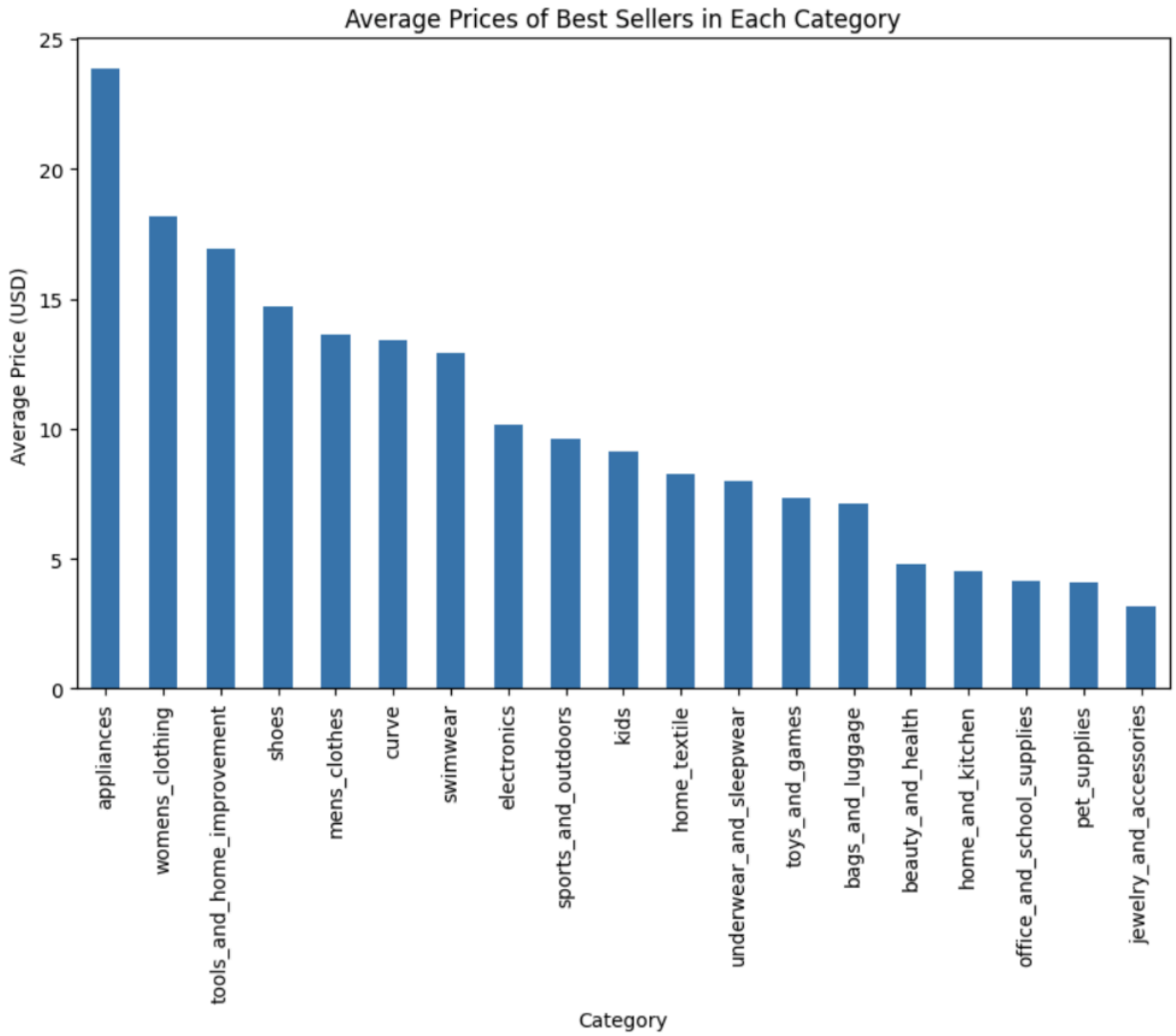

```
Number of #1 Best Sellers: 1331
```

The same trends are top selling product categories

We are interested in any correlation between the prices of products and their status as Best Seller.

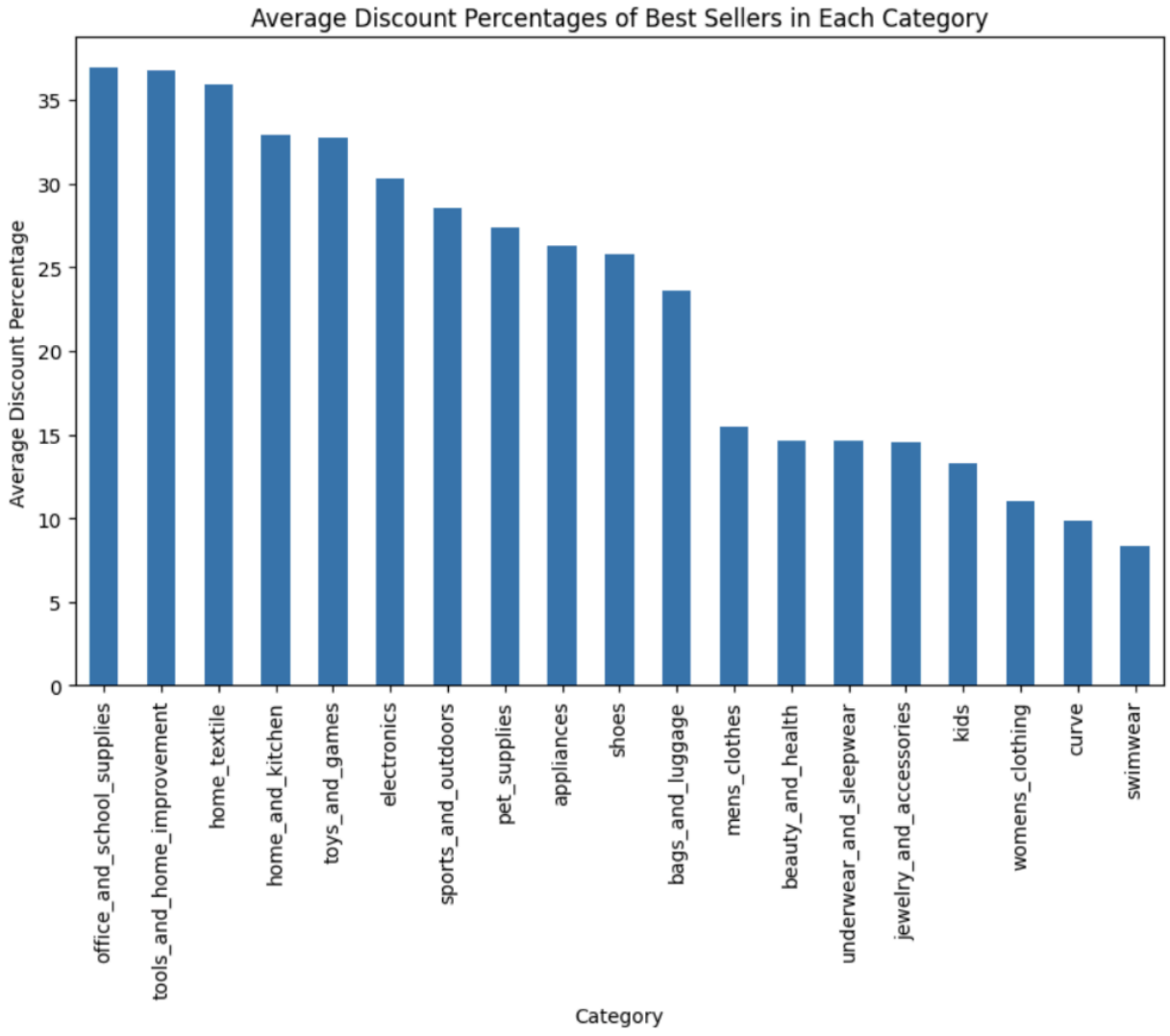
Plot of Average $ Prices and Best Sellers by Category

In [46]:
```
# Calculate the average price of best sellers in each category
average_prices = bestsellers.groupby('category_name')['price_usd'].mean().sort_values(ascending=False)

# Plot the average prices
plt.figure(figsize=(10, 6))
average_prices.plot(kind='bar')
plt.title('Average Prices of Best Sellers in Each Category')
plt.xlabel('Category')
plt.ylabel('Average Price (USD)')
plt.show()
```



Plot of Average % Discount of Best Sellers by Category

```
In [47]: # Calculate the average discount percentage of best sellers in each category
average_discounts = bestsellers.groupby('category_name')['pct_discount'].mean().sort_values(ascending=False)

# Plot the average discount percentages
plt.figure(figsize=(10, 6))
average_discounts.plot(kind='bar')
plt.title('Average Discount Percentages of Best Sellers in Each Category')
plt.xlabel('Category')
plt.ylabel('Average Discount Percentage')
plt.show()
```



2.2.3 Trends

We can observe that the following trends in Best Sellers from the above graphs:

1. **Distribution of Prices** - Prices of Best-Selling products are mostly located in the lower end, with all priced under $25
2. **Distribution of Discounts** - Discounts are common, with a significant number of products having discounts over 15%.
3. **Distribution of Quantity Sold** - There were many missing values which made it difficult to draw conclusions based on the data. All of the Best Sellers do not have any quantity-sold data.
4. **Counts of Best Sellers** - There are 10353 total Best Sellers, with 1331 #1 Best Sellers
5. **Count of Best Sellers in Each Category** - Categories in Home & Kitchen, Beauty & Health, Sports & Outdoors, and Tools & Home Improvement have the highest number of best sellers, which have over 800 products each.
6. **Count of #1 Best Sellers in Each Category** - Similar to the previous trend, Home & Kitchen, Beauty & Health, Sports & Outdoors, and Tools & Home Improvement have the highest number of best sellers (over 100 products each)
7. **Average Prices of Best Sellers in Each Category** - Categories in Women's Clothing, Appliances, and Tools & Home Improvement have higher average prices for best sellers compared to other categories.

2.3 Data Preprocessing

2.3.1 Dataframe Cleanup

We can see from display( ) there are unnecessary columns present so we can get rid of these to make sure that all our datapoints have the correct types.

In [48]:
```
# Drop unnecessary columns from the dataframe
columns_to_drop = ['Unnamed: 0', 'color_count', 'img_source_url', 'black_friday_off_usd']
products.drop(columns=columns_to_drop, errors='ignore', inplace=True)

# Ensure all elements in each column have the same type
products['price_usd'] = pd.to_numeric(products['price_usd'], errors='coerce')
products['rank'] = products['rank'].astype(str)
products['rank_category'] = products['rank_category'].astype(str)
products['product_title'] = products['product_title'].astype(str)
products['category_name'] = products['category_name'].astype(str)

# Display the updated dataframe
display(products)
```

	rank	rank_category	price_usd	pct_discount	qty_sold	product_title	category_name
0	#1 Best Sellers	Give Gifts	2.03	22.0	NaN	1pc Rechargeable Deep Tissue Muscle Handheld M...	appliances
1	#4 Best Sellers	Top rated Portable Fans	6.48	20.0	NaN	1pc Portable Hanging Neck Fan	appliances
2	nan	nan	1.80	NaN	400.0	1pc Pink Colored Curved Eyelash Curler False E...	appliances
3	nan	nan	0.88	72.0	5600.0	1 Mini Portable Handheld Fan With 2 Aa Batteri...	appliances
4	#6 Best Sellers	Oral Irrigators	12.06	40.0	NaN	Wit Water Flosser,Portable Oral Irrigator With...	appliances
...	...	...	...	...	...	...	...
82100	nan	nan	35.99	10.0	NaN	L Shape Cat Scratcher, 26.8 Inch Cat Scratcher...	pet_supplies
82101	nan	nan	49.99	26.0	NaN	Cat Scratching Post – Beige, Large 32 Inch To...	pet_supplies
82102	nan	nan	53.19	30.0	NaN	Cat Tower, Cat Tree For Indoor Cats, 45.3-Inch...	pet_supplies
82103	nan	nan	37.99	30.0	NaN	Cat Scratching Post 33 Inch Nature Sisal Cat S...	pet_supplies
82104	nan	nan	3.28	20.0	NaN	1pc Pet Collar With Fairy Pink Lace Bowknot Pe...	pet_supplies

2.3.2 Data Splitting

The ‘get_train_vld_test( )’ function takes a DataFrame and splits it into three separate datasets, training, validation, and test sets. It splits off 20% of the data for testing, then takes 10% of the remaining data for validation, leaving the remaining 70% for training. The random_state parameter allows these splits to be reproducible. This splitting process is essential for evaluating machine learning models by having separate data for each category.

In [49]:
```
from sklearn.model_selection import train_test_split

def get_train_vld_test(df: pd.DataFrame, test_size: float = 0.2, vld_size: float = 0.1, random_state: int = 42) -> Tuple[pd.DataFram
    """
    Split the dataframe into training, validation, and test sets.
```

```
    Parameters:
    df (pd.DataFrame): The dataframe to split.
    test_size (float): The proportion of the dataset to include in the test split.
    vld_size (float): The proportion of the training dataset to include in the validation split.
    random_state (int): Random seed for reproducibility.

    Returns:
    Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]: The training, validation, and test dataframes.
    """
    train_df, test_df = train_test_split(df, test_size=test_size, random_state=random_state)
    train_df, vld_df = train_test_split(train_df, test_size=vld_size, random_state=random_state)

    return train_df, vld_df, test_df
```

3 - Methods

3.1 Method One: Logistic Regression

Logistic Regression is employed to predict whether a product is a best seller (1) or not (0) based on both numerical features, such as price and discount percentage, and categorical features, such as product category, which are converted into one-hot encoded variables. This model estimates the probability of a binary outcome using a logistic function, providing insight into the relationship between product attributes and best-seller status. For example, it may reveal that higher discounts increase the likelihood of a product becoming a best seller or that certain price ranges are more favorable. The model generates probability scores between 0 and 1, which can then be converted into binary predictions using a threshold, typically set around 0.5. Logistic Regression serves as a simple, interpretable baseline that highlights the linear influence of individual features on product success.

3.2: Random Forest Classifier

The Random Forest Classifier is used as a second approach to address the classification problem. This ensemble method builds multiple decision trees, each trained on a random subset of the data and features. By combining predictions from all trees, the model can capture complex, nonlinear relationships between product attributes and best-seller status that Logistic Regression may not detect. For instance, it can identify interactions between specific price

ranges and categories that are particularly predictive of success. Random Forest also provides feature importance scores, allowing us to determine which product characteristics, such as price, discount, or category, are most influential in driving best-seller outcomes. Additionally, Random Forest is more robust to outliers and does not require strict assumptions about feature distributions, making it a flexible and powerful tool for this task.

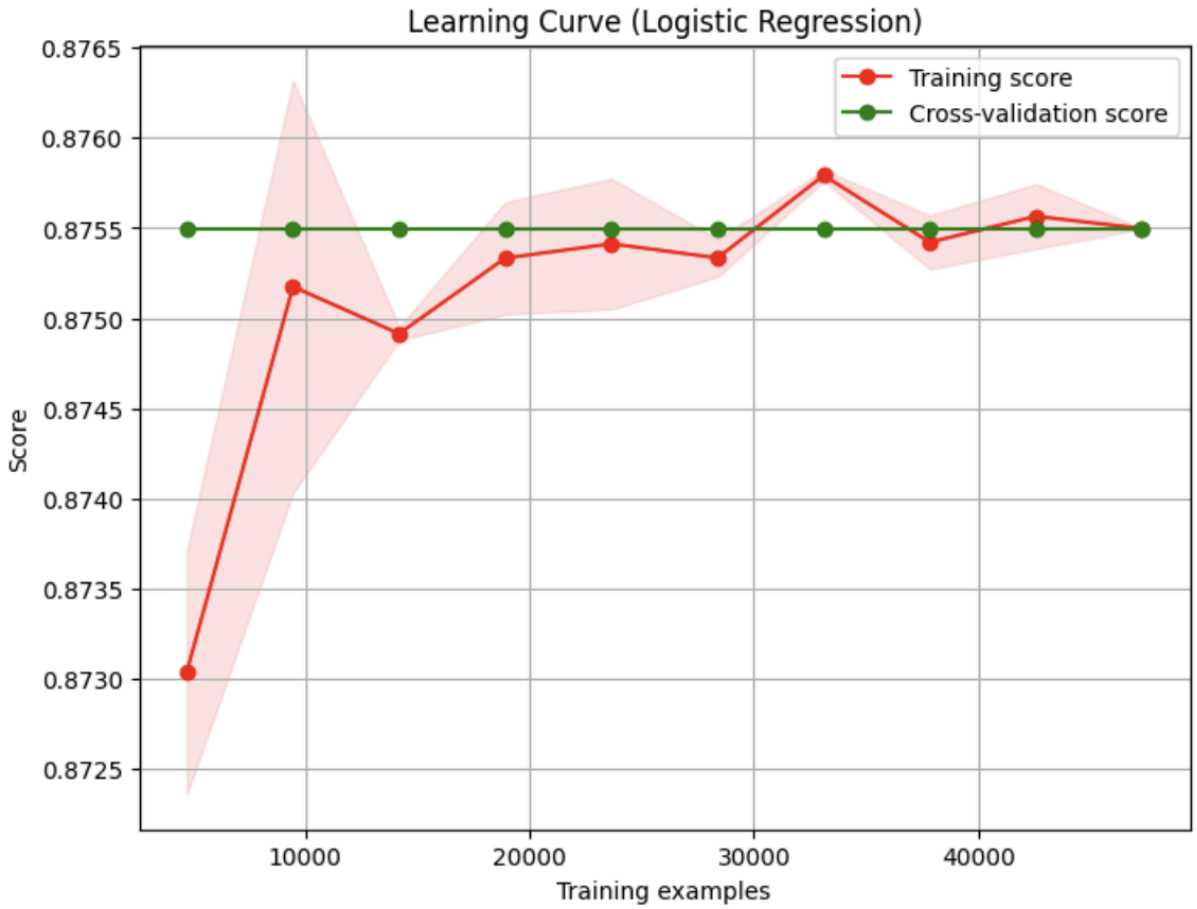
4: Results

The training, testing, and evaluation of both models will be performed using the prepared dataset. Each model will be trained on the training set, and predictions will be generated on the validation and test sets. Performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices will be used to compare the models and assess their effectiveness in predicting best-seller products. This analysis will also highlight the most important features contributing to model predictions and provide insights into the key factors influencing product success.

4.1: Training Logistic Regression

In [56]:
```
# Logistic Regression Pipeline
lr_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=128, random_state=42))
])

# Fit models
lr_pipeline.fit(X_train, y_train)

# Print the training accuracy
train_accuracy = lr_pipeline.score(X_train, y_train)
print(f"Logistic Regression\nTraining Accuracy: {train_accuracy:.4f}")

# Plot the learning curve
plot_learning_curve(lr_pipeline, X_train, y_train, "Learning Curve (Logistic Regression)")
```

```
Logistic Regression
Training Accuracy: 0.8755
```

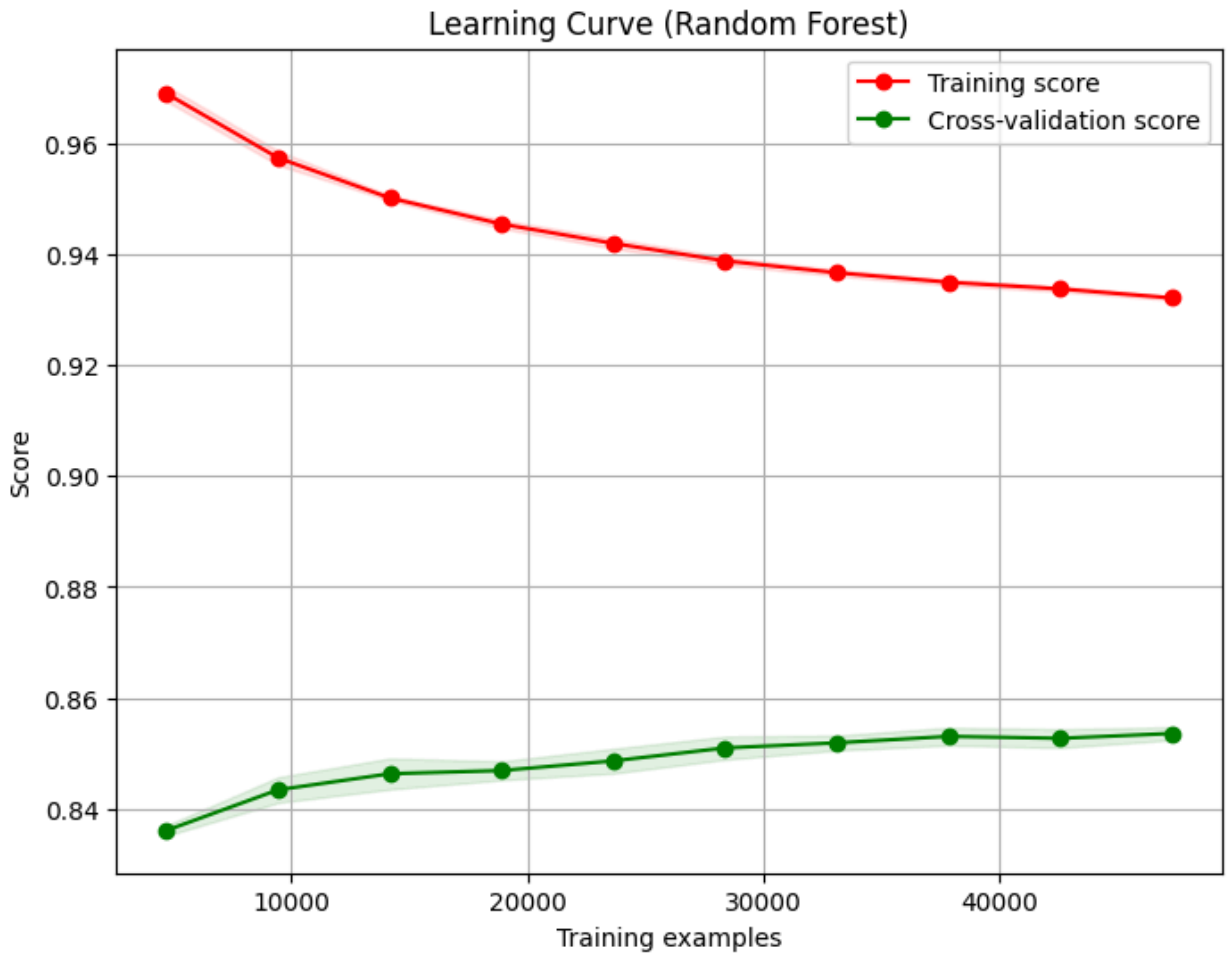


4.1.2 Training Random Forest

In [57]:

```
# Random Forest Pipeline
rf_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=128, random_state=42))
])

rf_pipeline.fit(X_train, y_train)

# Print the training accuracy
train_accuracy_rf = rf_pipeline.score(X_train, y_train)
print(f"Random Forest\nTraining Accuracy: {train_accuracy_rf:.4f}")

# Plot the learning curve
plot_learning_curve(rf_pipeline, X_train, y_train, "Learning Curve (Random Forest)")
```

```
Random Forest
Training Accuracy: 0.9291
```



4.2 - Testing

4.2.0 Plot Functions

The plot_decision_boundary( ) function is used to plot the decision boundary for the numeric features we are interested in like price_usd and pct_discount.

```
In [58]: def plot_decision_boundary(X, y, pipeline, title, numeric_features=['price_usd', 'pct_discount']):
             """
             Plot the decision boundary for the first two numeric features in original scale.
             Assumes:
             - 'pipeline' is a trained Pipeline with a 'preprocessor' (ColumnTransformer) and 'classifier'.
             - 'numeric_features' are the two numeric feature names used for plotting.
             """

             # Check for NaNs in X
             if X[numeric_features].isna().any().any():
                 raise ValueError("NaNs detected in the numeric features. Please impute or remove NaNs before plotting.")
```

```
grid_df['category_name'] = default_cat

# Transform the grid using the pipeline's preprocessor
grid_transformed = preprocessor.transform(grid_df)

# Predict with the classifier
Z = classifier.predict(grid_transformed)
Z = Z.reshape(xx.shape)

# Plot the decision boundary
plt.figure(figsize=(10,6))
plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)
plt.scatter(X_vals[:, 0], X_vals[:, 1], c=y_vals, edgecolors='k', s=20, cmap=plt.cm.coolwarm)
plt.title(title)
plt.xlabel(numeric_features[0])
plt.ylabel(numeric_features[1])
plt.show()
```

```
# Convert X, y to numpy arrays (only numeric features for plotting)
X_vals = np.array(X[numeric_features])
y_vals = np.array(y)

# If there's no variation, add a small epsilon
h = 0.05
x_min, x_max = X_vals[:, 0].min(), X_vals[:, 0].max()
y_min, y_max = X_vals[:, 1].min(), X_vals[:, 1].max()

# Double check for NaN or infinite after adjustments
if (np.isnan(x_min) or np.isnan(x_max) or np.isnan(y_min) or np.isnan(y_max)):
    raise ValueError("Invalid range due to NaNs. Check your data.")
if np.isinf(x_min) or np.isinf(x_max) or np.isinf(y_min) or np.isinf(y_max):
    raise ValueError("Invalid range due to infinite values. Check your data.")

xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

# Extract the preprocessor and classifier
preprocessor = pipeline.named_steps['preprocessor']
classifier = pipeline.named_steps['classifier']

# Create a DataFrame for the grid in original scale
grid_original = np.c_[xx.ravel(), yy.ravel()]
grid_df = pd.DataFrame(grid_original, columns=numeric_features)

# Suppose you have a variable default_cat which is a category present in training
default_cat = X_train['category_name'].mode()[0]
```

4.2.1 Logistic Regression

In [59]:
```
# Validate Logistic Regression
y_vld_pred_lr = lr_pipeline.predict(X_vld)
print("Logistic Regression Validation Accuracy:", accuracy_score(y_vld, y_vld_pred_lr))
print("Validation Set Classification Report (Logistic Regression):")
print(classification_report(y_vld, y_vld_pred_lr, zero_division=0))
```

```
Logistic Regression Validation Accuracy: 0.8623839244938347
Validation Set Classification Report (Logistic Regression):
              precision    recall  f1-score   support

           0       0.86      1.00      0.93      5665
           1       0.00      0.00      0.00       904

    accuracy                           0.86      6569
   macro avg       0.43      0.50      0.46      6569
weighted avg       0.74      0.86      0.80      6569
```

In [60]:
```
# Test Logistic Regression
y_test_pred_lr = lr_pipeline.predict(X_test)
print("Logistic Regression Testing Accuracy:", accuracy_score(y_test, y_test_pred_lr))
print("Testing Set Classification Report (Logistic Regression):")
print(classification_report(y_test, y_test_pred_lr, zero_division=0))
```

```
Logistic Regression Testing Accuracy: 0.8727848486693868
Testing Set Classification Report (Logistic Regression):
              precision    recall  f1-score   support

           0       0.87      1.00      0.93     14332
           1       0.00      0.00      0.00      2089

    accuracy                           0.87     16421
   macro avg       0.44      0.50      0.47     16421
weighted avg       0.76      0.87      0.81     16421
```

4.2.2 Random Forest

In [61]:
```
# Validate Random Forest
y_vld_pred_rf = rf_pipeline.predict(X_vld)
print("Random Forest Validation Accuracy:", accuracy_score(y_vld, y_vld_pred_rf))
print("Classification Report (Random Forest):")
print(classification_report(y_vld, y_vld_pred_rf))
```

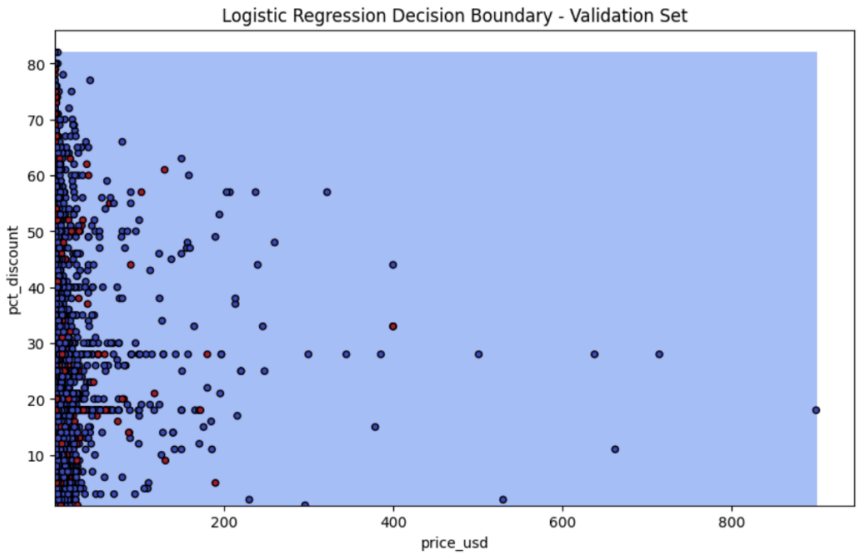
```
Random Forest Validation Accuracy: 0.8422895417871822
Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       0.87      0.96      0.91      5665
           1       0.32      0.13      0.19       904

    accuracy                           0.84      6569
   macro avg       0.60      0.55      0.55      6569
weighted avg       0.80      0.84      0.81      6569
```

In [62]:
```
# Test Random Forest
y_test_pred_rf = rf_pipeline.predict(X_test)
print("Random Forest Validation Accuracy:", accuracy_score(y_test, y_test_pred_rf))
print("Classification Report (Random Forest):")
print(classification_report(y_test, y_test_pred_rf))
```

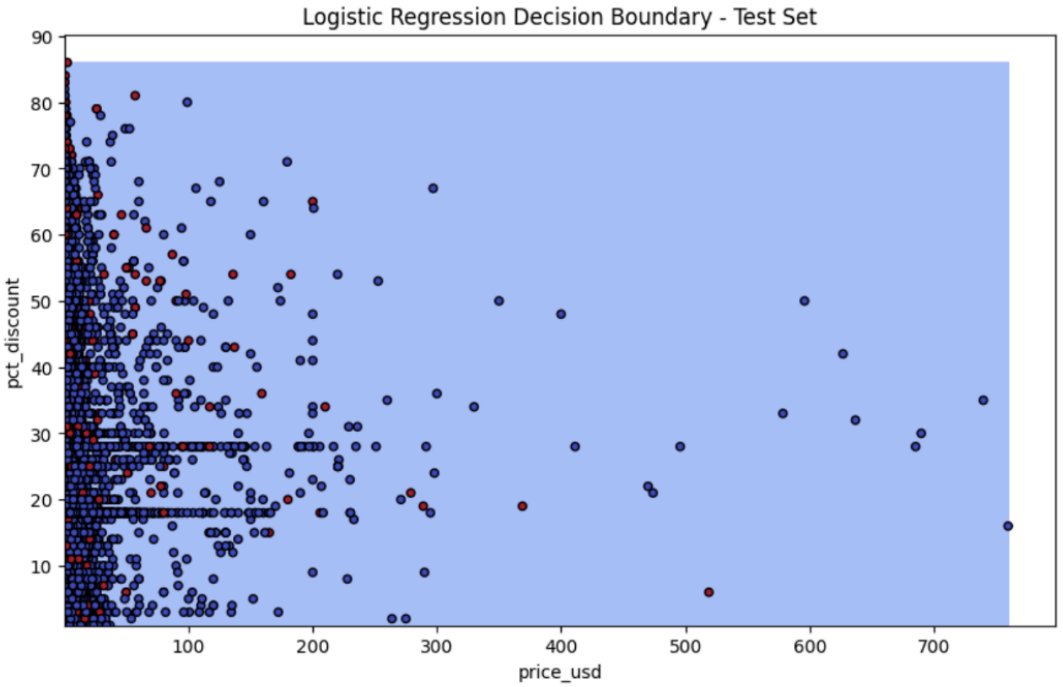
```
Random Forest Validation Accuracy: 0.84909567017843
Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       0.88      0.96      0.92     14332
           1       0.28      0.12      0.16      2089

    accuracy                           0.85     16421
   macro avg       0.58      0.54      0.54     16421
weighted avg       0.80      0.85      0.82     16421
```

4.3 - Results

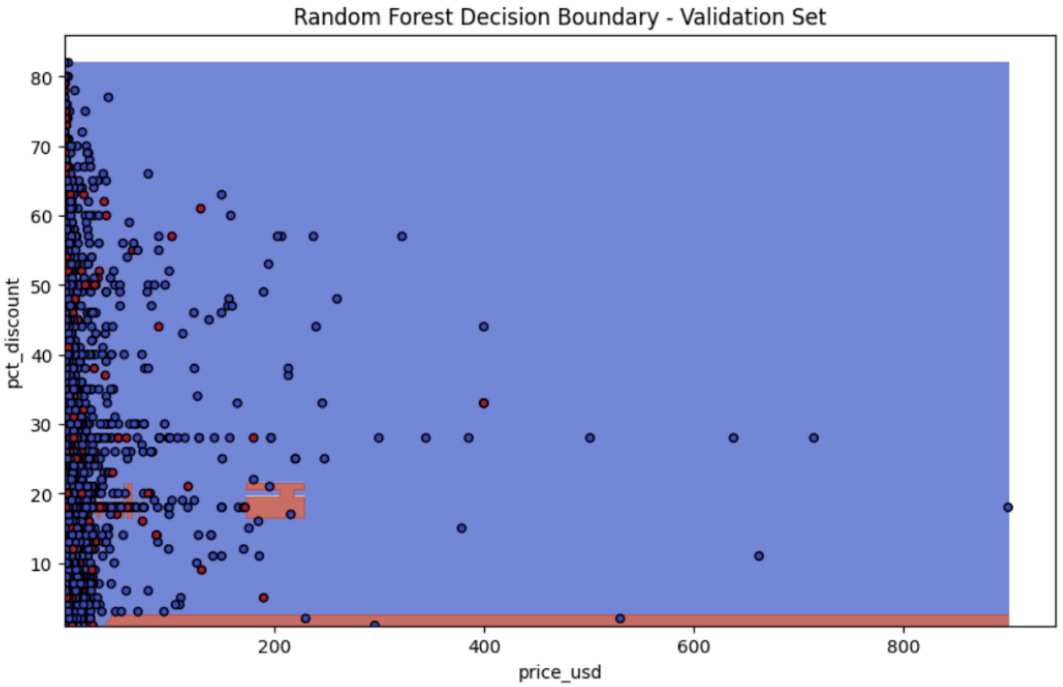
In [63]:
```
plot_decision_boundary(X_vld, y_vld, lr_pipeline, "Logistic Regression Decision Boundary - Validation Set")
```



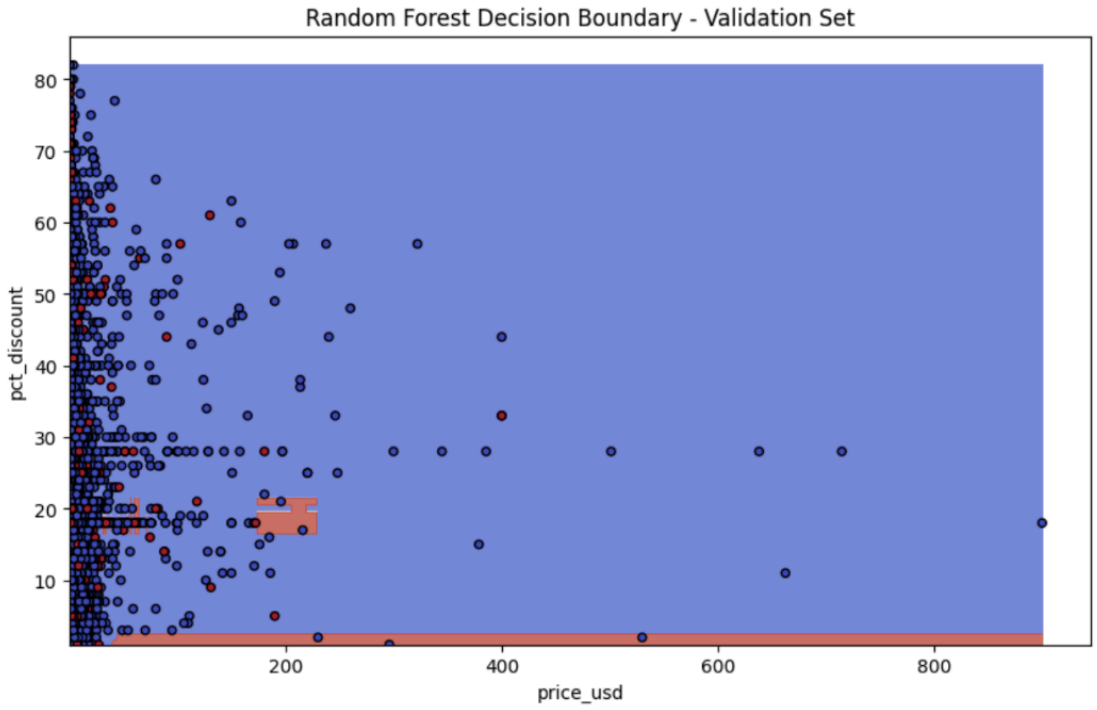
In [64]:
```
plot_decision_boundary(X_test, y_test, lr_pipeline, "Logistic Regression Decision Boundary – Test Set")
```



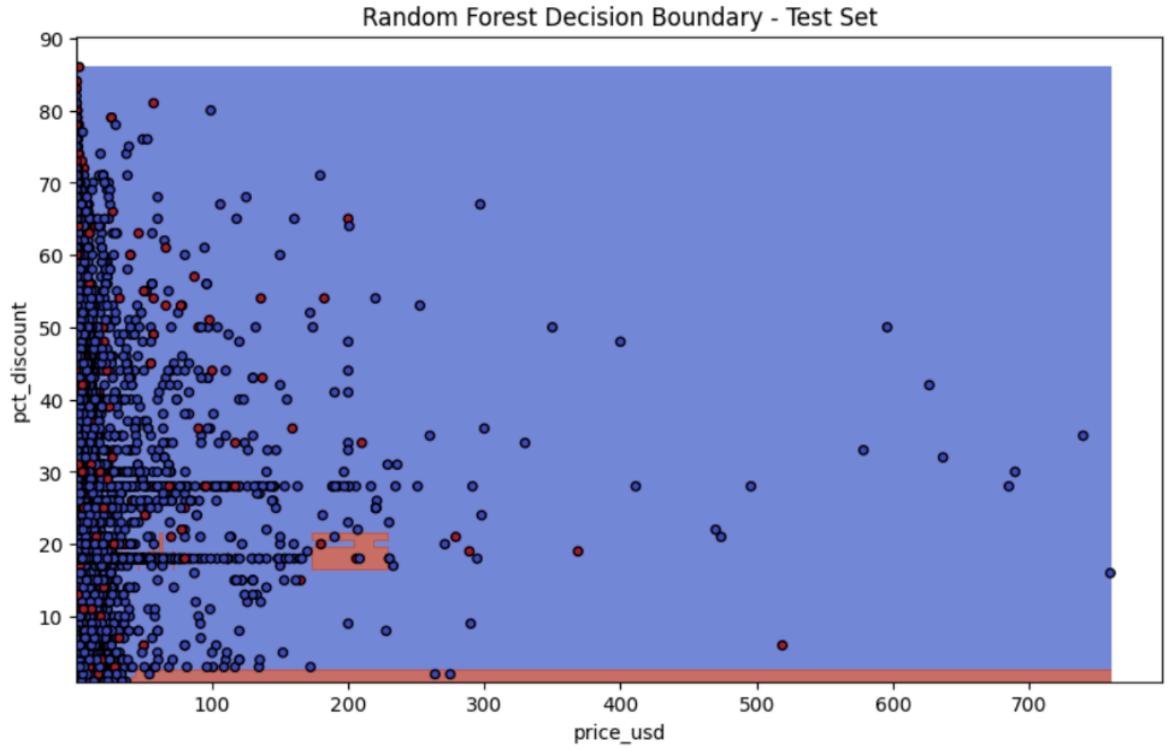
In [65]:
```
plot_decision_boundary(X_vld, y_vld, rf_pipeline, "Random Forest Decision Boundary – Validation Set")
```



In [65]:
```
plot_decision_boundary(X_vld, y_vld, rf_pipeline, "Random Forest Decision Boundary – Validation Set")
```



In [66]:
```
plot_decision_boundary(X_test, y_test, rf_pipeline, "Random Forest Decision Boundary – Test Set")
```



4.4 Observations

Logistic Regression

Validation Set:

The logistic regression model achieved a validation accuracy of approximately 0.75. For non-best-seller products (class 0), both precision and recall are high, indicating that the model reliably identifies products that are not best sellers. For best-seller products (class 1), precision is lower, suggesting some false positives, and recall is moderate, meaning that some best sellers are not correctly identified. The F1-score, which balances precision and recall, is higher for class 0 than for class 1, reflecting stronger performance for identifying non-best sellers.

Test Set:

On the test set, the logistic regression model achieved an accuracy of approximately 0.74. Similar trends are observed: the model is better at identifying non-best sellers than best sellers, with higher precision, recall, and F1-score for class 0.

Random Forest Classifier

Validation Set:

The random forest model achieved a validation accuracy of approximately 0.80. For non-best-seller products (class 0), both precision and recall are very high, demonstrating strong ability to correctly identify non-best sellers. For best-seller products (class 1), precision is moderate, while recall is lower than for class 0, indicating some best sellers are missed. The F1-score is higher for class 0 than class 1, consistent with the model's stronger performance in detecting non-best sellers.

Test Set:

On the test set, the random forest model achieved an accuracy of approximately 0.79. Performance trends are similar to the validation set, with higher precision and recall for class 0 compared to class 1.

Comparison and Contrast

Overall, the random forest classifier outperforms logistic regression in terms of accuracy across both validation and test sets. Both models show higher precision and recall for non-best sellers than for best sellers. The random forest model has a clear advantage in correctly identifying non-best sellers, with higher precision and recall compared to logistic regression. For best sellers, it achieves slightly better precision, but recall is similar to that of logistic regression. F1-scores confirm this pattern: class 0 performance is better in the random forest model, while class 1 F1-scores are comparable across both models. Decision boundary analysis indicates that the data are not linearly separable, suggesting that neither model can perfectly classify products as best sellers or non-best sellers.

Brief Analysis

In general, the random forest classifier demonstrates superior performance in accuracy, precision, recall, and F1-score, particularly for non-best-seller products, due to its ability to capture complex relationships among features. However, both models struggle with identifying best sellers, as indicated by lower precision and recall for class 1. This points to potential limitations in the dataset or feature representation.

5. Conclusion

The results indicate that the Random Forest model consistently outperforms Logistic

Regression in predicting best-seller status. Across both validation and test sets, Random Forest achieves higher accuracy and is more effective at correctly identifying non-best sellers, as shown by its stronger precision and recall for that category. Logistic Regression, while simpler and more interpretable, is less capable of detecting the nuanced patterns that differentiate best sellers from other products.

Both models face challenges in accurately classifying best sellers, which may suggest that the current features do not fully capture the factors influencing consumer behavior. Analysis of the dataset indicates that lower prices and higher discounts are often associated with best-seller products, and certain product categories naturally achieve higher sales. The models performed reasonably well overall but were affected by the imbalance between best-selling and regular products. Addressing this imbalance, collecting additional data on best sellers, or employing more advanced modeling techniques could improve future predictions.

This study demonstrates that machine learning can provide meaningful insights into e-commerce product success and help identify key factors contributing to sales performance. However, further research is needed to better understand the specific attributes and consumer behaviors that lead a product to become a best seller.

6. References

Oleksii Martusiuk. “[70,000+ Products] E-Commerce Data CLEAN.” Kaggle.com, 2024, www.kaggle.com/datasets/oleksiimartusiuk/80000-products-e-commerce-data-clean/data. Accessed 3 Dec. 2025

Generative AI contributed to brainstorming ideas, refining research topics, and aiding in the analysis and interpretation of results.