



"ONLINE SHOPPERS PURCHASE INTENTION **PREDICTIVE ANALYSIS** USING MACHINE LEARNING"

PROJECT REPORT

By

SAGAR SS

DATA ANALYST

UNDER THE GUIDANCE AND SUPERVISION OF

JIMCY



Kochi, Kerala 682025

INTRODUCTION

In today's competitive e-commerce landscape, understanding customer purchase behavior is critical for online retailers. Millions of visitors browse websites daily, but only a fraction convert into actual buyers. Predicting whether a visitor is likely to make a purchase allows businesses to improve customer experience, personalize marketing strategies, and reduce cart abandonment rates.

Traditional analytics often fail to capture the complex interactions between user behavior, page visits, bounce rates, and seasonal factors. This project leverages machine learning techniques to predict online shoppers' purchase intentions based on behavioral and demographic patterns. By analyzing historical browsing sessions, including page categories visited, visit duration, bounce rates, and visitor type, machine learning models can uncover hidden patterns that drive purchase decisions.

The outcome of this project enables e-commerce companies to identify high-potential customers, implement targeted promotions, and optimize website design for improved conversion rates.

ABSTRACT

This project aims to develop an Online Shoppers Purchase Intention Predictive Model using machine learning techniques. The primary goal is to analyze how visitors interact with an e-commerce website and predict whether a user will complete a purchase (Revenue = 1) or not (Revenue = 0).

The dataset consists of user session data such as administrative, informational, and product-related page visits, session duration, bounce/exit rates, page values, visitor type, month of visit, and weekend indicators. The dependent variable is Revenue, while the independent variables include session-based behavioral and categorical attributes.

We applied multiple classification algorithms — Logistic Regression, K-Nearest Neighbors (KNN), Decision Tree, and Random Forest — with preprocessing steps such as encoding categorical data, feature scaling, and hyperparameter tuning using GridSearchCV.

Among all models, Random Forest delivered the highest accuracy (~89–92%), making it the most reliable in predicting customer purchase behavior. The findings offer actionable insights for e-commerce stakeholders to increase conversions, personalized shopping experiences, and allocate marketing resources more effectively.

OBJECTIVE

To develop a Purchase Intention Predictive Analysis Model using machine learning for e-commerce websites.

Specific objectives include:

- Develop an Optimized Machine Learning Model:
 - Implement and fine-tune classification algorithms (Logistic Regression, Decision Tree, Random Forest, KNN).
 - Apply hyperparameter tuning using GridSearchCV to improve predictive performance.
 - Evaluate model performance using accuracy, precision, recall, F1-score, and ROC-AUC.

- Feature Importance & Impact Analysis:
 - Identify key behavioral features affecting purchase intent (Product Related pages, Page Values, ExitRates, etc.).
 - Analyze how factors like visitor type (Returning vs New) and seasonal trends (Month, Weekend) influence conversions.
- Predictive Insights for Stakeholders:
 - Enable businesses to recognize high-intent buyers in real time.
 - Provide actionable strategies to reduce bounce rates and increase engagement.
 - Support decision-making for targeted marketing campaigns and personalized recommendations.

DATA ANALYSIS PROCEDURE

1. Import Essential Libraries

- Pandas, NumPy for data handling.
- Matplotlib, Seaborn for visualization.
- Scikit-learn for machine learning models and evaluation.

2. Load the Data

- Read the dataset using `pandas.read_csv()`.
- Display structure using `.info()`, `.head()`, `.describe()`.

3.Check for Missing Values and Clean Data

- Use `.isnull().sum()` to detect missing values.
- Handle missing values appropriately (if any).

4.Explore and Visualize Data

- Analyze distribution of categorical features (Visitor Type, Month, Weekend).
- Plot histograms, box plots, and heatmaps to understand correlations.

5.Assign Dependent and Independent Variables

- **Dependent Variable:** Revenue (Purchase: Yes/No).
- **Independent Variables:** Page visit counts, Bounce Rates, ExitRates, Page Values, Month, Visitor Type, Weekend.

6.Encode Categorical Variables

- Apply Label Encoding or One-Hot Encoding to categorical features (Month, Visitor Type).

7.Feature Scaling

- Scale numerical features (e.g., Bounce Rates, ExitRates) using StandardScaler for models like KNN.

8.Split the Data (Train-Test Split)

- Divide into training (80%) and testing (20%).

9. Train Models Using Classification Algorithms

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- K-Nearest Neighbors (KNN)

10. Optimize Model Performance

- Use GridSearchCV for hyperparameter tuning.

11. Evaluate Model Performance

- Evaluate accuracy, precision, recall, F1-score, and ROC-AUC.
- Compare confusion matrices for all models.

Generate Insights

- Identify which features most strongly influence purchase decisions.
- Recommend strategies for improving conversion rates.

ABOUT DATASET

This dataset, collected from UCI Machine Learning Repository, contains online shoppers' behavior sessions from an e-commerce platform.

Dataset Columns:

1. Administrative / Administrative_Duration – Pages related to site administration and their duration.
2. Informational / Informational_Duration – Informational page visits and time spent.
3. Product Related / ProductRelated_Duration – Product-related pages visited and duration.
4. Bounce Rates – Probability of a user leaving after the first page.
5. ExitRates – Probability of exiting the site from a given page.
6. Page Values – Value of pages based on conversion likelihood.
7. Special Day – Proximity to special shopping days (e.g., Black Friday).

8. Month – Month of visit (categorical).
9. Operating Systems, Browser, Region, TrafficType – Technical info.
10. Visitor Type – Returning vs New Visitor.
11. Weekend – Boolean indicating if the session was on a weekend.
12. Revenue – Target variable (1 = Purchase, 0 = No Purchase)

METHODOLOGY

1. Libraries Used

- Pandas, NumPy – Data handling.
- Matplotlib, Seaborn – Visualization.
- Scikit-learn – Preprocessing, model training, and evaluation.

2. Machine Learning Models Applied

- **Logistic Regression:** For baseline binary classification.
- **K-Nearest Neighbors (KNN):** Distance-based classifier.
- **Decision Tree:** Rule-based classification.
- **Random Forest:** Ensemble of decision trees for robust predictions.

3. Preprocessing Techniques

- One-Hot Encoding for categorical features.
- StandardScaler for normalization.
- Train-Test split to evaluate performance.

4. Hyperparameter Tuning

- GridSearchCV to optimize `n_neighbors` (KNN), `max_depth` (Decision Tree), and `n_estimators` (Random Forest).

DATA INTERPRETATION

Dependent Variable: Revenue (purchase intent).

Independent Variables:

- Session behavior (pages visited, durations, bounce/exit rates).
- Visitor attributes (Visitor Type, Weekend).
- Seasonal effects (Month, SpecialDay).

Key Observations:

- **Random Forest** achieved the best accuracy (~89–92%).
- **Important Features:** Product Related, Page Values, and ExitRates were the strongest predictors of purchase.
- Returning Visitors had higher chances of making purchases than New Visitors.
- Seasonal effects (November, December) showed higher conversion rates due to holidays.

SUGGESTIONS

1. Business Strategy Optimization

- Focus on improving Product Related pages and Page Values, as they directly influence purchase intent.
- Reduce ExitRates by optimizing checkout and payment pages.

2. Personalized Marketing

- Target Returning Visitors with special promotions and loyalty programs.
- Provide offers during peak shopping months (November, December).

3. Customer Retention

- Use real-time predictions to trigger interventions (e.g., pop-ups, chatbots, or discounts) when a high-intent buyer is about to leave.

4. Website Design & User Experience

- Minimize bounce rates by improving landing page engagement.
- Provide easy navigation to product-related sections.

5. Predictive Analytics for Growth

- Continuously monitor customer behavior and retrain models to adapt to changing trends.

CONCLUSION

This project demonstrated how machine learning can effectively predict online shoppers' purchase intentions by analyzing behavioral, categorical, and temporal features. Among the models tested, Random Forest Classifier emerged as the most accurate and reliable, with ~89–92% prediction accuracy.

The findings highlight that Product Related pages, Page Values, and Exit Rates significantly impact purchase likelihood. Business strategies such as improving product page engagement, targeting returning visitors, and offering time-sensitive promotions can enhance conversion rates.

By implementing this predictive model, e-commerce businesses can make data-driven decisions, personalize user experiences, and increase overall revenue. Looking ahead, incorporating real-time prediction engines and integrating deep learning techniques could further improve accuracy and adaptability to evolving customer behaviors.

Appendix - I

25

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	Visits
	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	0.0	Feb	1	1	1	1	Return
	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	0.000000	0.0	Feb	2	2	1	2	Return
	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	0.0	Feb	4	1	9	3	Return
	0	0.0	0	0.0	2	2.666667	0.050000	0.140000	0.000000	0.0	Feb	3	2	2	4	Return
	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	0.000000	0.0	Feb	3	3	1	4	Return

25	3	145.0	0	0.0	53	1783.791667	0.007143	0.029031	12.241717	0.0	Dec	4	6	1	1	Return
26	0	0.0	0	0.0	5	465.750000	0.000000	0.021333	0.000000	0.0	Nov	3	2	1	8	Return
27	0	0.0	0	0.0	6	184.250000	0.083333	0.086667	0.000000	0.0	Nov	3	2	1	13	Return
28	4	75.0	0	0.0	15	346.000000	0.000000	0.021053	0.000000	0.0	Nov	2	2	3	11	Return
29	0	0.0	0	0.0	3	21.250000	0.000000	0.066667	0.000000	0.0	Nov	3	2	1	2	Return

20 rows x 18 columns

Appendix - II

```
# Step 1: Libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,  
classification_report
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
```

**Load Dataset**

```
df = pd.read_csv("/content/online_shoppers_intention.csv")
print("Shape :",df.shape)
df.head()
df.tail()
print(df.info())
print(df.describe())
```

**Exploratory Data Analysis(EDA)**

Distribution of Purchase Decision

```
sns.countplot(data=df, x='Revenue', palette='Set2')
plt.title("Distribution of Purchase Decision")
plt.show()
```

Distribution of Key Numeric Features

```
num_cols = ['Administrative', 'Informational', 'Product Related',
            'Bounce Rates', 'ExitRates', 'Page Values']
```

```
df[num_cols].hist(bins=20, figsize=(12,8), color='skyblue')
plt.suptitle("Distributions of Key Numeric Features")
plt.show()
```

Correlation Heatmap

```
plt.figure(figsize=(10,6))  
sns.heatmap(df[num_cols + ['Revenue']].corr(), annot=True, cmap='coolwarm')  
plt.title("Correlation Heatmap")  
plt.show()
```

Data Preprocessing

```
print(df.isnull().sum())
```

Encode Categorical Variables

```
cat_cols = ['Month','Visitor Type','Weekend']  
le = LabelEncoder()  
for col in cat_cols:  
    df[col] = le.fit_transform(df[col])
```

Split Features & Target

```
X = df.drop('Revenue', axis=1)  
y = df['Revenue']
```

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.2, random_state=42, stratify=y)
```

Scale Numeric Features

```
scaler = StandardScaler()  
X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

Model Training

i)KNeighborsClassifier

```
knn = KNeighborsClassifier(n_neighbors=5)  
knn.fit(X_train, y_train)  
y_pred_knn = knn.predict(X_test)  
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
```

ii)DecisionTreeClassifier

```
dt = DecisionTreeClassifier(random_state=42)  
dt.fit(X_train, y_train)  
y_pred_dt = dt.predict(X_test)  
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
```

iii)RandomForestClassifier

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)  
rf.fit(X_train, y_train)  
y_pred_rf = rf.predict(X_test)  
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

****iv)LogisticRegression****

```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
```

```
models = {
    "KNN": KNeighborsClassifier(n_neighbors=5), # default k=5, can tune later
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Logistic Regression": LogisticRegression(max_iter=2000)
}
```

Dictionary to store accuracies

```
accuracies = {}
```

Loop over models

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

```
acc = accuracy_score(y_test, y_pred)
accuracies[name] = acc # store accuracy
```

```
print(f" ♦ {name} Results ♦ ")
print("Accuracy:", acc)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("=="*50)
```

Hyperparameter Tuning

```
knn = KNeighborsClassifier()
param_knn = {
    'n_neighbors': [3,5,7,9],
    'weights': ['uniform','distance'],
    'metric': ['euclidean','manhattan']
}

grid_knn = GridSearchCV(knn, param_grid=param_knn, cv=5, scoring='accuracy')
grid_knn.fit(X_train, y_train)

knn_best = grid_knn.best_estimator_
knn_pred = knn_best.predict(X_test)

print("Best KNN Params:", grid_knn.best_params_)
print("KNN Accuracy:", accuracy_score(y_test, knn_pred))

dt = DecisionTreeClassifier(random_state=42)
param_dt = {
    'max_depth': [5,10,15,None],
    'min_samples_split': [2,5,10],
    'criterion': ['gini','entropy']
}
```

```
grid_dt = GridSearchCV(dt, param_grid=param_dt, cv=5, scoring='accuracy')
grid_dt.fit(X_train, y_train)
```

```
dt_best = grid_dt.best_estimator_
dt_pred = dt_best.predict(X_test)
```

```
print("Best DT Params:", grid_dt.best_params_)
print("Decision Tree Accuracy:", accuracy_score(y_test, dt_pred))
```

```
rf = RandomForestClassifier(random_state=42)
param_rf = {
    'n_estimators': [100,200],
    'max_depth': [5,10,15,None],
    'min_samples_split': [2,5,10]
}
```

```
grid_rf = GridSearchCV(rf, param_grid=param_rf, cv=5, scoring='accuracy')
grid_rf.fit(X_train, y_train)
```

```
rf_best = grid_rf.best_estimator_
rf_pred = rf_best.predict(X_test)
```

```
print("Best RF Params:", grid_rf.best_params_)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
```

```
lr = LogisticRegression(max_iter=2000)
param_lr = {
    'C': [0.01,0.1,1,10],
    'solver': ['lbfgs','liblinear']
}

grid_lr = GridSearchCV(lr, param_grid=param_lr, cv=5, scoring='accuracy')
grid_lr.fit(X_train, y_train)

lr_best = grid_lr.best_estimator_
lr_pred = lr_best.predict(X_test)

print("Best LR Params:", grid_lr.best_params_)
print("Logistic Regression Accuracy:", accuracy_score(y_test, lr_pred))

results = {
    "KNN": accuracy_score(y_test, knn_pred),
    "Decision Tree": accuracy_score(y_test, dt_pred),
    "Random Forest": accuracy_score(y_test, rf_pred),
    "Logistic Regression": accuracy_score(y_test, lr_pred)
}

print("Model Accuracies:", results)
```

##Evaluate Tuned Models##

```
def plot_cm(y_true, y_pred, title):  
    cm = confusion_matrix(y_true, y_pred)  
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  
    plt.xlabel("Predicted")  
    plt.ylabel("Actual")  
    plt.title(f"Confusion Matrix - {title}")  
    plt.show()
```

KNN

```
print("KNN Classification Report")  
print(classification_report(y_test, knn_pred))  
plot_cm(y_test, knn_pred, "KNN")
```

Decision Tree

```
print("Decision Tree Classification Report")  
print(classification_report(y_test, dt_pred))  
plot_cm(y_test, dt_pred, "Decision Tree")
```

Random Forest

```
print("Random Forest Classification Report")  
print(classification_report(y_test, rf_pred))  
plot_cm(y_test, rf_pred, "Random Forest")
```

Logistic Regression

```
print("Logistic Regression Classification Report")  
print(classification_report(y_test, lr_pred))  
plot_cm(y_test, lr_pred, "Logistic Regression")
```

Accuracy Comparison Chart

Identify best model

```
best_model_name = max(results, key=results.get)
```

```
best_model_acc = results[best_model_name]
```

```
print(f"\n✅ Best Model: {best_model_name} with Accuracy = {best_model_acc:.4f}")
```

Bar chart highlighting best model

```
plt.figure(figsize=(7,4))
```

```
bars = sns.barplot(x=list(results.keys()), y=list(results.values()), palette="magma")
```

```
for i, model in enumerate(results.keys()):
```

```
    if model == best_model_name:
```

```
        bars.patches[i].set_color('green')
```

```
plt.ylabel("Accuracy")
```

```
plt.title("Model Comparison After Tuning (Best Highlighted in Green colour)")
```

```
plt.show()
```

REFERENCES : [UCI Machine Learning Repository.](#)