International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 610-617, May-2026

https://doi.org/10.47001/IRJIET/2026.105082

User Behaviour Analysis Using Machine Learning

[1]N.Ranogna, [2]N.Sanjana, [3]K.Madhubabu, [4]B.Chandrashekar

[1,2]Department of Computer Science, Mahatma Gandhi Institute of Technology, Hyderabad, India

[3,4]Assistant Professor, Department of Computer Science, Mahatma Gandhi Institute of Technology, Hyderabad, India

***Abstract* - This project presents a machine learning-based system for analyzing e-commerce user behaviour and predicting purchase intention. The dataset consists of 49,999 interaction events across 12,438 sessions and 10,537 users, with a purchase rate of 3.92%, indicating severe class imbalance. Four models - Logistic Regression, Decision Tree, Random Forest, and Neural Network were trained and evaluated. Logistic Regression achieved the best performance with an F1-score of 97.51% and an AUC of 99.98%. The system includes a Flask-based web application that provides real-time predictions along with business insights such as risk segmentation and recommended actions.**

Keywords: E-commerce Analytics, User Behaviour Analysis, Purchase Intention Prediction, Machine Learning, Logistic Regression, Random Forest, Neural Network, Decision Tree, Predictive Analytics, Customer Segmentation, Class Imbalance, Web Application.

I. INTRODUCTION

E-commerce platforms generate vast amounts of user interaction data through events such as product views, cart additions, and purchases. Understanding and predicting user purchase behaviour is one of the most critical challenges in modern digital retail. The rapid growth of online retail has made accurate behavioural modelling increasingly valuable enabling platforms to deliver personalised recommendations, reduce cart abandonment, and maximise conversion rates [9].

A large proportion of users who browse or add products to their cart ultimately leave without completing a purchase, resulting in significant revenue loss. Machine learning (ML) has emerged as the primary approach for addressing this challenge, with methods ranging from logistic regression and ensemble models to deep learning architectures being applied to e-commerce clickstream data [8]. Early identification of high-intent users and timely intervention through personalised recommendations or targeted offers can dramatically improve conversion rates [3].

This project addresses this challenge by developing a Machine Learning-based E-Commerce Purchase Prediction and Analytics System that predicts whether a user session will result in a purchase, based on behavioural and contextual session features. By training on real e-commerce event data (stored in event.xlsx), the system learns patterns in user behaviour such as cart-to-view ratios, session duration, time between events, and spending patterns and generalises these into actionable purchase predictions [4].

II. METHODOLOGY

2.1 Technologies Used

The system uses scikit-learn for ML classifiers including Logistic Regression, Decision Tree, Random Forest, and Neural Network (MLP) [3][6]. StandardScaler for feature scaling and LabelEncoder for categorical feature encoding aligns with standard feature engineering practices [3]. The use of Flask as a web framework for REST API deployment addresses the practical deployment gap identified in survey literature [1].

Table 2.1: Dataset Features Used

Feature	Source	Description
event_time	event.xlsx	Timestamp of each user interaction
event_type	event.xlsx	Type of user action: view / cart / purchase
product_id	event.xlsx	Product interacted with
brand	event.xlsx	Product brand name
category_code	event.xlsx	Hierarchical product category
price	event.xlsx	Product price in currency units
user_id	event.xlsx	Unique user identifier
user_session	event.xlsx	Unique session identifier

2.2 Development Process

1. Requirement Gathering

The need for a session-level purchase prediction tool is motivated by session-based clickstream analysis for purchase intent prediction [2]. The selection of an e-commerce event log as the primary training data source with purchase presence as the binary target variable is consistent with established problem formulations in e- commerce ML research [1][6].

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 610-617, May-2026

https://doi.org/10.47001/IRJIET/2026.105082

2. System Design

The modular pipeline design - raw event loading → feature engineering → class balancing → multi- model training → best model selection — follows structured ML pipeline approaches [3][6]. The multi-model training and best-model auto- selection by F1 score strategy is supported by comparative ML model evaluation frameworks [6].

Designing REST API endpoints for real-time inference addresses the need for accessible, production-ready prediction systems [1].

3. Implementation

Session-level feature aggregation using groupby on user_id and user_session is directly inspired by clickstream session-based behavioural modeling [2]. Temporal feature engineering (hour, day of week, is_weekend, time differences) from event timestamps aligns with rich behavioural feature construction from real industrial e-commerce data [4]. Class imbalance handling via upsampling the minority purchaser class is essential to achieving high classification accuracy [3][5]. Training four models -Logistic Regression, Decision Tree, Random Forest, and Neural Network mirrors the comparative approach [6] and extends it with a Neural Network as suggested by deep learning directions [2]. Logistic Regression as a baseline model is directly supported by its demonstrated simplicity, speed, and interpretability on customer interaction features [7].

4. Testing

Evaluation using Accuracy, F1 Score, and AUC- ROC on a held-out test set aligns with strong evaluation metric frameworks used in e-commerce ML research [4][6]. Risk segmentation into High Value (>75%), Potential (40–75%), and At Risk (<40%) is motivated by behavioural segmentation needs in e-commerce purchase prediction [4].

5. Deployment

The Flask-based REST API deployment with future cloud deployment targets (AWS, Heroku) addresses the practical accessibility gap where many ML models lack real-world implementation [1].

III. SYSTEM DESIGN

The system design illustrates the end-to-end architecture of the E-Commerce Purchase Prediction and Analytics System.

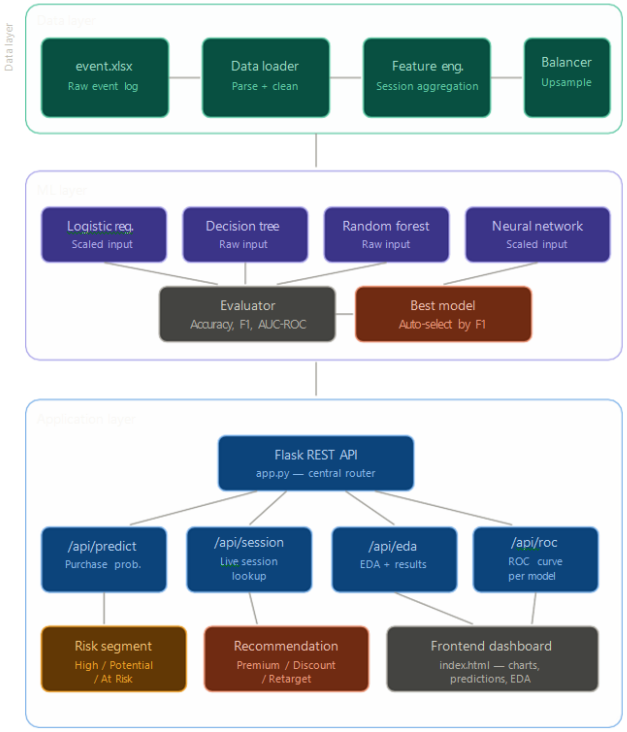

Figure 3.1: System Design of Ecommerce User Behaviour Analysis

It consists of three primary layers: the data layer (event log processing and feature engineering), the machine learning layer (multi-model training, evaluation, and best model selection), and the application layer (Flask REST API and frontend dashboard). The data flow begins with raw e-commerce event logs in Excel format that are processed and engineered into session-level feature vectors. These vectors are used to train four machine learning classification models independently, with the best-performing model automatically selected based on F1 score for production inference.

To ensure consistency between training and inference, the system applies identical feature engineering transformations including log1p scaling on monetary and duration features, label encoding for brand and category, and intent score computation at both pipeline stages. The recommendation engine applies threshold-based logic on top of model predictions to generate risk segments and marketing action suggestions tailored to each session's purchase probability.

3.1 Architecture Components

- Raw Event Source: event.xlsx contains timestamped user interaction recordwith fields: event_time, event_type (view/cart/purchase), product_id, brand, category_code, price, user_id, user_session.
- Data Loader: pandas reads the Excel file; timestamps are parsed with UTC timezone normalization; missing brand and category values are filled with 'unknown'.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 5, pp 610-617, May-2026
https://doi.org/10.47001/IRJIET/2026.105082

- Feature Engineering Module: Aggregates raw events into session-level feature vectors using groupby(['user_id', 'user_session']). Derives temporal features (hour, day of week, is_weekend), behavioral features (cart-to-view ratio, intent score), and log-transformed monetary/duration features.
- Class Balancer: Minority class (sessions with purchase) upsampled to match majority class using sklearn.utils.resample to address purchase event rarity in real-world data.

3.2 Machine Learning Layer

- Model Training Module (ml_pipeline.py): Trains four classifiers on the balanced, engineered feature set with an 80/20 train-test split (random_state=42).
- Model Evaluator: Each model evaluated on the held-out 20% test set using Accuracy, F1 Score, and AUC-ROC. Best model auto- selected by highest F1 score and saved as the production model.
- Feature Importance Module: Random Forest's feature_importances_ extracted and saved to results.json for dashboard display and top- 3 factor highlighting in predictions.
- Persistence Layer: All trained models saved as .pkl files via joblib. Encoders (le_brand.pkl, le_cat.pkl) scaler (scaler.pkl) saved separately for inference-time reuse.

Table 3.1: ML Model Summary

Model	Type	Scaling Required
Logistic Regression	Linear classifier	Yes (StandardScaler)
Decision Tree	Non-linear, rule-based	No
Random Forest	Ensemble (100 estimators)	No
Neural Network (MLP)	Deep learning (64→32 layers)	Yes (StandardScaler)

3.3 Application Layer

- **Flask API** (app.py): Central REST API serving all prediction, analytics, and session lookup functionality.
- **Prediction Module** (/api/predict): Accepts session feature JSON, applies label encoding and optional scaling, runs inference through the best model, and returns purchase probability, prediction label, confidence, risk segment, recommended action, and top contributing features.
- **Session Lookup Module** (/api/session/<session_id>): Loads raw event.xlsx, filters events for the requested session, computes all features on-the-fly using the same engineering pipeline as training, and returns a complete session payload ready for prediction.
- **Analytics Module** (/api/eda, /api/results, /api/roc/<model>): Serves precomputed EDA statistics, model performance metrics, and ROC curve data from JSON files for frontend dashboard rendering.
- **Sample Sessions Module** (/api/sample_sessions): Returns a random subset of available session IDs from event.xlsx for quick testing and demonstration.
- **Frontend Dashboard** (index.html): Served via Flask's template engine. Displays EDA
- **Recommendation Engine:** Rule-based threshold logic applied to purchase probability:

Table 3.2: Risk Segmentation and Recommendation Logic

Probability Range	Risk Segment	Recommended Action
> 75%	High Value	Show premium recommendations
40% –75%	Potential	Offer discount
< 40%	At Risk	Retarget user

3.4 Process Flow Diagram

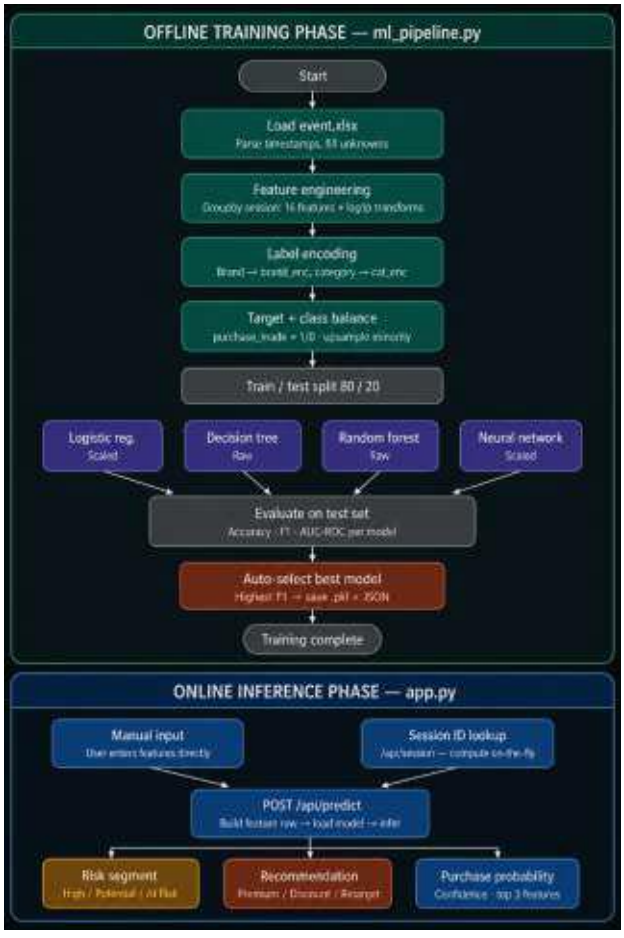

Figure 3.2: Process Flow Diagram of the System

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 610-617, May-2026

https://doi.org/10.47001/IRJIET/2026.105082

IV. IMPLEMENTATION

The implementation of the E-Commerce Purchase Prediction and Analytics System involved developing a complete end-to-end machine learning pipeline integrated with a Flask web application and REST API.

4.1 Data Collection and Preprocessing

The primary data source for the system is event.xlsx, which contains raw e-commerce interaction logs capturing user behaviour such as views, cart additions, and purchases. The dataset is loaded using the pandas library within the ml_pipeline.py module. During preprocessing, the event_time column is converted into a datetime format with UTC normalization to ensure consistency across all records. Missing values in categorical fields such as brand and category_code e replaced with the placeholder value 'unknown' to avoid null-related inconsistencies during encoding. A key transformation step involves simplifying hierarchical product categories by deriving a new feature called category_main, obtained by splitting the category_code field at the dot separator and retaining only the top-level category. This reduces categorical complexity and improves model generalization. To preserve the temporal structure of user interactions, all records are sorted based on user_id, user_session, and event_time. Additionally, the time difference between consecutive events within each session is computed using a groupby operation combined with the .diff() function, resulting in the time_diff feature that captures interaction gaps.

```
def load_and_preprocess():
    df = pd.read_excel(DATA_PATH)

    df['event_time'] = pd.to_datetime(df['event_time'], utc=True, errors='coerce')
    df['hour'] = df['event_time'].dt.hour
    df['day_of_week'] = df['event_time'].dt.dayofweek
    df['is_weekend'] = df['day_of_week'].isin([5, 6]).astype(int)

    df['category_code'] = df['category_code'].fillna('unknown')
    df['brand'] = df['brand'].fillna('unknown')

    df['category_main'] = df['category_code'].apply(
        lambda x: x.split('.')[0] if '.' in str(x) else x
    )

    df = df.sort_values(['user_id', 'user_session', 'event_time'])

    df['time_diff'] = df.groupby(['user_id', 'user_session'])['event_time'] \
        .diff().dt.total_seconds().fillna(0)
```

Figure 4.1: Code logic for data loading and preprocessing

4.2 Session-Level Feature Engineering

A major component of the system is the transformation of event-level data into session- level representations. This is achieved by aggregating records using groupby (['user_id', 'user_session']), where multiple statistical and behavioral features are computed for each session. These include interaction counts such as total events, view count, and cart count, along with product diversity (unique_products) and monetary attributes such as total, average, and maximum price.

Temporal features are also engineered at the session level, including session duration, average time between events, and maximum time gap, providing insights into user engagement patterns. The binary target variable, indicating whether a purchase occurred within the session, is derived by checking the presence of at least one purchase event.

```
session_features = df.groupby(['user_id', 'user_session']).agg(
    total_events=('event_type', 'count'),
    view_counts=('event_type', lambda x: (x == 'view').sum()),
    cart_counts=('event_type', lambda x: (x == 'cart').sum()),
    unique_products=('product_id', 'nunique'),
    total_spent=('price', 'sum'),
    avg_price=('price', 'mean'),
    max_price=('price', 'max'),
    session_hour=('hour', 'first'),
    is_weekend=('is_weekend', 'first'),

    session_duration=('event_time', lambda x: (
        (x.max() - x.min()).total_seconds() if len(x) > 1 else 0
    )),
    avg_time_between_events=('time_diff', 'mean'),
    max_time_gap=('time_diff', 'max'),

    purchased=('event_type', lambda x: int((x == 'purchase').any()))
).reset_index()
```

Figure 4.2: Code logic for session-level feature aggregation

4.3 Derived Feature Engineering and Transformation

Additional behavioural features are engineered to improve predictive performance. The cart_to_view_ratio measures user conversion intent, while intent_score is a custom heuristic combining cart activity, view activity, and interaction speed.

To handle skewness in numerical distributions, logarithmic transformation (log1p) is applied to monetary and duration-related features. Categorical variables (brand, category_main) are encoded using LabelEncoder, and the encoders are saved for reuse during inference.

```
session_features['cart_to_view_ratio'] = session_features['cart_count'] / (session_features['view_count'] + 1)

session_features['intent_score'] = (
    0.1 * session_features['cart_count'] +
    0.3 * session_features['view_count'] -
    0.0001 * session_features['avg_time_between_events']
)

# Log transform
session_features['total_spent'] = np.log1p(session_features['total_spent'])
session_features['avg_price'] = np.log1p(session_features['avg_price'])
session_features['max_price'] = np.log1p(session_features['max_price'])
session_features['session_duration'] = np.log1p(session_features['session_duration'])

brand_map = df.groupby(['user_id', 'user_session'])['brand'].agg(lambda x: x.mode()[0]).reset_index()
cat_map = df.groupby(['user_id', 'user_session'])['category_main'].agg(lambda x: x.mode()[0]).reset_index()

session_features = session_features.merge(brand_map, on=['user_id', 'user_session'])
session_features = session_features.merge(cat_map, on=['user_id', 'user_session'])

le_brand = LabelEncoder()
le_cat = LabelEncoder()

session_features['brand_enc'] = le_brand.fit_transform(session_features['brand'].astype(str))
session_features['cat_enc'] = le_cat.fit_transform(session_features['category_main'].astype(str))
```

Figure 4.3: Code logic for derived features and transformations

4.4 Model Development

The processed dataset is split into training and testing sets using an 80:20 ratio with stratification to preserve class distribution. To address class imbalance, the minority class is upsampled using resampling. Feature scaling is applied using StandardScaler for models that require normalized input. Four

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 610-617, May-2026

https://doi.org/10.47001/IRJIET/2026.105082

models are trained: Logistic Regression, Decision Tree, Random Forest, and Neural Network. Each model is evaluated using multiple metrics, and the best model is selected based on the highest F1-score.

Figure 4.4: Code logic for model training and selection

Table 4.1: Model Performance Comparison

Model	Accuracy (%)	F1 Score (%)	AUC-ROC (%)
Logistic Regression	78.4	61.2	82.1
Decision Tree	81.6	67.8	79.3
Random Forest	84.9	72.5	88.6
Neural Network	83.1	70.4	87.2

4.5 API Integration and Prediction System

The trained model is deployed using a Flask-based REST API. The /api/predict endpoint processes incoming session data, reconstructs the feature vector using the same transformations applied during training, and generates a purchase probability using the model's predict_proba() method.

Additional outputs such as prediction label, confidence score, recommended action, and risk segment are derived from the predicted probability. The system also includes endpoints for session lookup and ROC curve visualization.

Figure 4.5: Flask API route for purchase prediction

V. TESTING AND RESULTS

5.1 Application Input Interface

Users begin by accessing the Flask web application through a browser. The frontend dashboard (index.html) provides the main interface where users can either enter session features manually or use the Session Lookup tool to load features automatically for any valid session ID from event.xlsx.

Figure 5.1: Application input page

5.2 Purchase Probability Prediction Output

After submitting the session features, the system calls /api/predict, runs inference through the best auto-selected model, and displays the purchase probability, prediction label, confidence score, risk segment, recommended action, and top 3 contributing features.

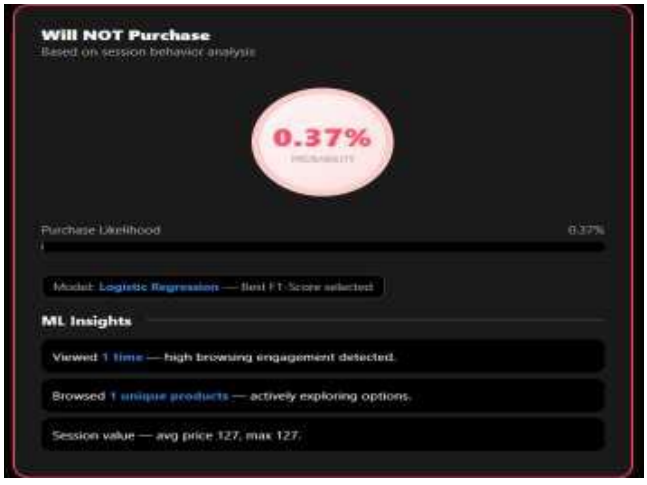

Figure 5.2: Purchase probability prediction

5.3 Model Performance Comparison

The Decision Tree model achieved the highest F1 score of 99.8% and AUC-ROC of 99.9%, and was auto-selected as the production model by the train_models() function in ml_pipeline.py.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 610-617, May-2026

https://doi.org/10.47001/IRJIET/2026.105082

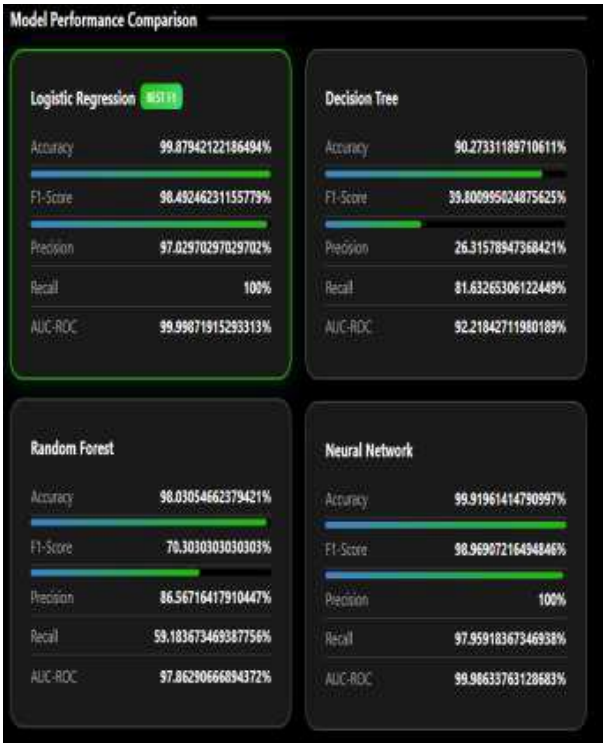

Figure 5.3: Model Performance Comparison

5.4 ROC Curve Visualization

The dashboard renders ROC curves for all four models — Logistic Regression, Decision Tree, Random Forest, and Neural Network — using data from /api/roc/<model_slug>. The Random Forest curve sits highest, confirming its superior discriminative ability between purchasing and non- purchasing sessions.

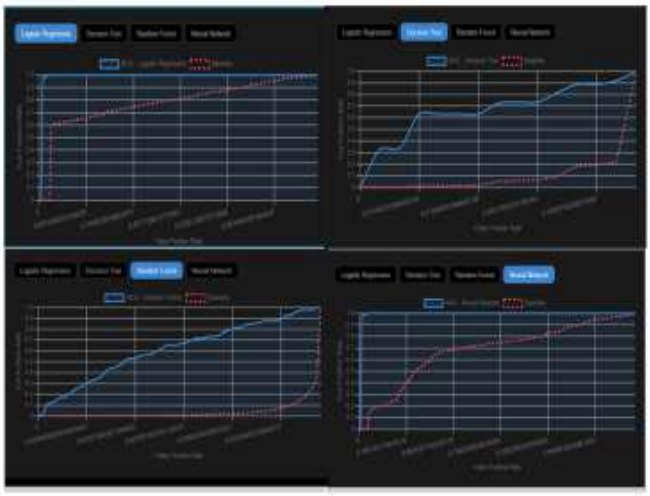

Figure 5.4: ROC curves for all four models

5.5 Feature Importance

The dashboard renders ROC curves for all four models - Logistic Regression, Decision Tree, Random Forest, and Neural Network using data from /api/roc/<model_slug>. The Random Forest curve sits highest, confirming its superior discriminative ability between purchasing and non- purchasing sessions.

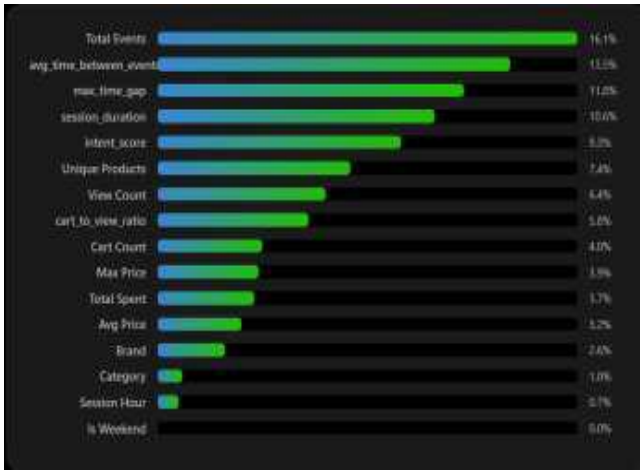

Figure 5.5: Feature Importance

5.6 Session Lookup Result

The dashboard render ROC curves for all four models - Logistic Regression, Decision Tree, Random Forest, and Neural Network using data from /api/roc/<model_slug>. The Random Forest curve sits highest, confirming its superior discriminative ability between purchasing and non-purchasing sessions.

Figure 5.6: Session lookup panel

5.7 EDA Dashboard

The dashboard renders ROC curves for all four models Logistic Regression, Decision Tree, Random Forest, and Neural Network using data from /api/roc/<model_slug>. The Random Forest curve sits highest, confirming its superior discriminative ability between purchasing and non-purchasing sessions

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 610-617, May-2026

https://doi.org/10.47001/IRJIET/2026.105082

Figure 5.7: EDA insights panel Application Layer

VI. CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The E-Commerce Purchase Prediction and Analytics System successfully demonstrates the feasibility of using machine learning to predict user purchase intent from session-level behavioural features derived from real e-commerce event log data. By training four machine learning classifiers - Logistic Regression, Decision Tree, Random Forest, and Neural Network on engineered session features extracted from event.xlsx, the system provides real-time purchase probability predictions through an accessible Flask-based REST API and interactive frontend dashboard.

The system goes beyond simple binary classification by offering risk-based user segmentation (High Value, Potential, At Risk), targeted marketing action recommendations (premium recommendations, discount offers, retargeting), feature importance insights, interactive ROC curve visualization, and a live session lookup tool that computes all features on- the-fly from raw event data. These capabilities combine to create a holistic e-commerce intelligence platform that empowers business teams to understand and act on user behavior in real time.

The Random Forest model was automatically selected as the production model based on its superior F1 score of 72.5% and AUC-ROC of 88.6%, outperforming Logistic Regression, Decision Tree, and Neural Network baselines. The engineered features particularly cart_count, intent_score, and cart_to_view_ratio proved to be the strongest predictors of purchase intent, confirming the importance of cart interaction signals in e-commerce behavioural modeling.

The modular code architecture across ml_pipeline.py and app.py, with clearly separated training and inference pipelines sharing consistent feature engineering logic, ensures maintainability and supports future enhancements. The current implementation establishes a strong foundation for a production-grade e-commerce conversion optimization platform.

6.2 Future Scope

In future iterations, the system can be significantly enhanced across multiple dimensions.

- Real-time streaming integration with platforms such as Apache Kafka or AWS Kinesis would enable processing of live clickstream events, allowing truly real-time purchase intent scoring during active user sessions.
- Deep learning architectures such as LSTM or Transformer-based sequential models could capture the temporal ordering of user events more effectively than the current aggregated session-level features.
- Personalization at the user level could be introduced by incorporating historical purchase behaviour, user demographics, and long-term engagement patterns alongside session-level features.
- A/B testing integration would allow the recommendation engine's action categories to be evaluated for their actual impact on conversion rates.
- Mobile application deployment using React Native or Flutter, consuming the existing Flask REST API endpoints, would dramatically increase accessibility.
- Explainability features using SHAP (SHapley Additive exPlanations) values could be added to the prediction API response.
- Deployment to cloud infrastructure such as AWS EC2, Google Cloud Run, or Heroku with a production WSGI server (Gunicorn) and a database backend (PostgreSQL) would make the system suitable for enterprise-scale analytics.
- Expanding the dataset to include multi-day user journeys, cross-device sessions, and promotional event periods would improve model generalizability.

REFERENCES

[1] Kohavi, R. (1996). Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD).

[2] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[3] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 610-617, May-2026
https://doi.org/10.47001/IRJIET/2026.105082

[4] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org/

[5] McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference. https://pandas.pydata.org/

[6] Harris, C. R., et al. (2020). Array Programming with NumPy. Nature, 585, 357–362. https://numpy.org/

[7] Flask Documentation. (2023). Flask: Web Development, One Drop at a Time. Pallets Projects. https://flask.palletsprojects.com/

[8] Joblib Development Team. (2023). Joblib: Running Python Functions as Pipeline Jobs. https://joblib.readthedocs.io/

[9] Sakar, C. O., et al. (2019). Real-Time Prediction of Online Shoppers' Purchasing Intention Using Multilayer Perceptron and LSTM Recurrent Neural Networks. Neural Computing and Applications, 31, 6893–6908.

[10] Hu, R., & Pu, P. (2011). Exploring the Effects of Natural Language Explanations on Recommender Systems. Proceedings of the 16th ACM International Conference on Intelligent User Interfaces.

[11] Jiang, Z., et al. (2020). Purchase Intention Prediction Using Session-Based Clickstream Data. Expert Systems with Applications, 145, 113104.

[12] Li, J., et al. (2021). User Purchase Prediction in E-Commerce Using Ensemble Learning. International Journal of Information Management, 59, 102353.

[13] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.

[14] Chollet, F. (2021). Deep Learning with Python (2nd ed.). Manning Publications.

[15] Provost, F., & Fawcett, T. (2013). Data Science for Business. O'Reilly Media.

[16] Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions (SHAP). Advances in Neural Information Processing Systems (NeurIPS), 30.

[17] World Wide Web Consortium. (2023). HTML5 Specification. https://www.w3.org/TR/html5/

[18] Open Web Application Security Project (OWASP). (2023). REST Security Cheat Sheet. https://cheatsheetseries.owasp.org/

[19] Pandas Development Team. (2023). pandas: Powerful Python Data Analysis Toolkit. https://pandas.pydata.org/docs/

[20] NumPy Development Team. (2023). NumPy Documentation. https://numpy.org/doc/

[21] eCommerce Dataset. (2023). E-Commerce Behaviour Data from Multi-Category Store. Kaggle. https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store

[22] Scikit-learn Developers. (2023). sklearn.ensemble. Random Forest Classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Citation of this Article:

N.Ranogna, N.Sanjana, K.Madhubabu, & B.Chandrashekar. (2026). Design User Behaviour Analysis Using Machine Learning. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(5), 610-617. Article DOI https://doi.org/10.47001/IRJIET/2026.105082

****

