KIET Institute of Technology
Ashish Gupta
CSE AI

- 2024 / 2025 -

PREDICTING ONLINE LEARNING COMPLETION USING MACHINE LEARNING.

🟦 2. INTRODUCTION

Introduction In today's fast-paced digital education landscape, understanding learner behavior is crucial for improving course engagement and completion rates. This project aims to predict whether a student will complete an online course based on activity logs such as videos watched, assignments submitted, and forum interactions. By using machine learning classification models, institutions can identify students who may need additional support.

🧪 3. Methodology

We utilized a classification approach for this supervised learning problem. The steps followed in this project were:

- Data Collection: Retrieved learner activity logs from a CSV file.
- Preprocessing: Converted categorical data into numerical format and handled missing values.
- Model Selection: Used Random Forest classifier to predict course completion.
- Evaluation: Measured model performance using accuracy, precision, recall, and confusion matrix visualization.

4. CODE

```python
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Preprocessing Function
def clean_text(text):
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    text = text.lower()
    text = re.sub(r'\s+', ' ', text).strip()
    return text
```

```python
# Load Dataset
df = pd.read_csv('/content/support_cases.csv')

# Clean Text Data
df['case_type'] = df['case_type'].apply(clean_text)

# Feature Extraction
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(df['case_type'])
y = df['category']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model 1: Multinomial Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_pred = nb_model.predict(X_test)

# Model 2: Logistic Regression
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)
```

```python
# Model Evaluation
print("Naive Bayes Classification Report:\n", classification_report(y_test, nb_pred))
print("Logistic Regression Classification Report:\n", classification_report(y_test, lr_pred))

# Confusion Matrix
conf_mat = confusion_matrix(y_test, lr_pred)
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()

# Classification Function
def classify_support_case(message):
    clean_msg = clean_text(message)
    vectorized_msg = vectorizer.transform([clean_msg])
    prediction = lr_model.predict(vectorized_msg)[0]
    confidence = max(lr_model.predict_proba(vectorized_msg)[0])
    return {"category": prediction, "confidence": round(confidence, 2)}
```

🙌 5. Credits

- Project By: Ashish Gupta
- Guided By: BIKKI KUMAR
- Tools Used: Python, Pandas, Scikit-learn, Matplotlib
- Data Source: online_learning.csv