School of Engineering & Technology

****

**Machine Learning and Pattern Recognition Lab Manual ENSP202**

**(2023-27)**

**Submitted by Amit Mohanty**

**Roll No. 2301730325**

**Submitted to : Mr. Kunal Rai**

#### University Vision & Mission

**Vision**

KR Mangalam University aspires to become an internationally recognized institution of higher learning through excellence in inter- disciplinary education, research and innovation, preparing socially responsible life-long learners contributing to nation building.

#### Mission

* Foster employability and entrepreneurship through futuristic curriculum and progressive pedagogy with cutting-edge technology.
* Instil notion of lifelong learning through stimulating research, Outcomes-based education, and innovative thinking.
* Integrate global needs and expectations through collaborative programs with premier universities, research centres, industries, and professional bodies.
* Enhance leadership qualities among the youth understanding ethical values and environmental realities.

#### School Vision & Mission

**Vision**

To excel in scientific and technical education through integrated teaching, research, and innovation

**Mission**

* **Creating** a unique and innovative learning experience to enhance quality in the domain of Engineering & Technology.
* **Promoting** Curricular, co-curricular and extracurricular activities that support overall personality development and lifelong learning, emphasizing character building and ethical behaviour.
* **Focusing** on employability through research, innovation and entrepreneurial mindset development.
* **Enhancing** collaborations with National and International organizations and institutions to develop cross-cultural understanding to adapt and thrive in the 21st century.

|  |  |  |
| --- | --- | --- |
| **EXPERIMENT NO.** | **EXPERIMENT TITLE** | **Page No.** |
| 1 | Understanding Credit Card Fraud Detection Models | 1 |
| 2 | Understanding Advertisement Budget Prediction Models | 6 |
| 3 | Understanding Diabetes Prediction Models | 8 |
| 4 | Understanding Credit Card Default Prediction Models | 10 |

# Experiment 1: Understanding Credit Card Fraud Detection Models

Introduction to Fraud Detection in Machine Learning

Credit card fraud detection is a crucial application of machine learning, as it helps financial institutions identify and prevent fraudulent activities in real-time. In this experiment, participants will be introduced to various techniques used in building fraud detection models, including data preprocessing, feature engineering, and machine learning algorithms.

Key Concepts in Credit Card Fraud Detection:

* Data Preprocessing: Cleaning and preparing data for training the model.
* Feature Engineering: Extracting important features from raw data to improve model performance.
* Machine Learning Algorithms: Algorithms like Decision Trees, Logistic Regression, and Random Forests used to predict fraud.
* Imbalanced Datasets: Handling the imbalance between fraudulent and non-fraudulent transactions.

### Objective

To understand the key concepts of credit card fraud detection. To explore the structure of fraud detection datasets.

To learn about different machine learning models and their applicability to fraud detection.

### Learning Outcomes

By the end of this experiment, participants will be able to:

1. Understand the importance of data preprocessing in fraud detection.
2. Apply feature engineering to enhance model performance.
3. Implement machine learning algorithms to detect fraud.
4. Evaluate model performance using metrics like precision, recall, and F1-score.

### Instructions for Conducting the Experiment

1. Preparation:
   * Ensure you have a Python environment set up (e.g., Jupyter Notebook).
   * Download a credit card fraud detection dataset (e.g., Kaggle's Credit Card Fraud Detection dataset).

import pandas as pd import numpy as np

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report from sklearn.impute import SimpleImputer

1. Data Preprocessing

# Load dataset (replace 'dataset.csv' with your file) df = pd.read\_csv('dataset.csv')

# Inspect for missing values

print("Missing values per column:\n", df.isnull().sum())

# Handle missing values (impute with mean for numerical data)

imputer = SimpleImputer(strategy='mean') df[df.select\_dtypes(include=[np.number]).columns] = imputer.fit\_transform(df.select\_dtypes(include=[np.number]))

# Identify and remove outliers using IQR method Q1 = df.quantile(0.25)

Q3 = df.quantile(0.75)

IQR = Q3 - Q1

df = df[~((df < (Q1 - 1.5 \* IQR)) | (df > (Q3 + 1.5 \* IQR))).any(axis=1)]

# Normalize numerical features scaler = StandardScaler()

df[df.select\_dtypes(include=[np.number]).columns] = scaler.fit\_transform(df.select\_dtypes(include=[np.number]))

1. Feature Engineering

# Select important features (modify based on dataset) selected\_features = ['transaction\_amount', 'time', 'location']

df = df[selected\_features + ['label']] # Assume 'label' is the target variable

# Create new feature: rolling average of transaction amount (window size = 3) df['rolling\_avg\_amount'] = df['transaction\_amount'].rolling(window=3, min\_periods=1).mean()

1. Model Selection

# Split data into features and target X = df.drop(columns=['label'])

y = df['label']

# Split into training and test sets (80% train, 20% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Choose a machine learning model (Random Forest)

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

1. Training the Model model.fit(X\_train, y\_train)

# Predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred) print(f"Accuracy: {accuracy:.4f}") print(classification\_report(y\_test, y\_pred))

Experiment 3: Understanding Baseball

Player Salary Prediction Models

Introduction to Salary Prediction in Machine Learning

Predicting baseball player salaries is an important application of machine learning in sports analytics. Teams and agents use these models to determine fair compensation based on player performance metrics. This experiment introduces regression techniques for salary prediction.

**Key Concepts:**

* Regression analysis for continuous value prediction
* Feature importance in sports analytics
* Performance metrics evaluation
* Handling skewed salary distributions

## Objective

To understand how machine learning can predict baseball player salaries based on performance statistics.

To explore the relationship between player statistics and compensation. To learn about regression models and their evaluation.

## Learning Outcomes

By the end of this experiment, participants will be able to:

1. Preprocess sports performance data for regression tasks
2. Implement linear regression and decision tree models
3. Evaluate model performance using RMSE and R-squared metrics
4. Interpret feature importance in salary prediction

## Instructions for Conducting the Experiment

1. Preparation:

import pandas as pd import numpy as np

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

1. Data Preprocessing:

df = pd.read\_csv('baseball\_stats.csv') df = df.dropna()

df = pd.get\_dummies(df, columns=['position'])

from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler()

numerical\_cols = ['batting\_avg', 'home\_runs', 'RBIs', 'years\_experience'] df[numerical\_cols] = scaler.fit\_transform(df[numerical\_cols])

1. Feature Engineering:

df['performance\_index'] = df['batting\_avg'] \* df['home\_runs']

features = ['batting\_avg', 'home\_runs', 'RBIs', 'years\_experience', 'position\_P', 'position\_C']

X = df[features] y = df['salary']

1. Model Training:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

dt\_model = DecisionTreeRegressor(max\_depth=5) dt\_model.fit(X\_train, y\_train)

1. Evaluation:

lr\_pred = lr\_model.predict(X\_test) dt\_pred = dt\_model.predict(X\_test)

print("Linear Regression:")

print(f"RMSE: {np.sqrt(mean\_squared\_error(y\_test, lr\_pred))}") print(f"R-squared: {r2\_score(y\_test, lr\_pred)}")

print("\nDecision Tree:")

print(f"RMSE: {np.sqrt(mean\_squared\_error(y\_test, dt\_pred))}") print(f"R-squared: {r2\_score(y\_test, dt\_pred)}")

# Experiment 2: Understanding Advertisement Budget Prediction Models

Introduction to Budget Prediction

Advertisement budget prediction helps marketing teams allocate resources efficiently. Machine learning models can predict optimal budgets based on historical campaign data and market conditions.

Key Concepts:

* Time series analysis for marketing data
* Multi-variable regression
* ROI prediction
* Marketing channel effectiveness

## Objective

To understand how machine learning predicts advertisement budgets. To analyze relationships between marketing channels and outcomes. To implement and compare different regression models.

## Learning Outcomes

By the end of this experiment, participants will be able to:

1. Preprocess marketing campaign data
2. Implement regression models for budget prediction
3. Evaluate model performance using business metrics
4. Interpret channel contribution to ROI

## Instructions for Conducting the Experiment

1. Preparation:

import pandas as pd

from sklearn.ensemble import RandomForestRegressor from sklearn.preprocessing import OneHotEncoder from sklearn.compose import ColumnTransformer

1. Data Preprocessing:

df = pd.read\_csv('ad\_campaigns.csv')

df['month'] = pd.to\_datetime(df['date']).dt.month df = df.drop('date', axis=1)

preprocessor = ColumnTransformer(

transformers=[('cat', OneHotEncoder(), ['channel\_type'])], remainder='passthrough')

X = preprocessor.fit\_transform(df.drop('budget', axis=1)) y = df['budget']

1. Model Training:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2) model = RandomForestRegressor(n\_estimators=100) model.fit(X\_train, y\_train)

1. Evaluation:

predictions = model.predict(X\_test)

print(f"RMSE: {np.sqrt(mean\_squared\_error(y\_test, predictions))}") print(f"Feature Importances: {model.feature\_importances\_}")

# Experiment 3: Understanding Diabetes Prediction Models

Introduction to Medical Diagnosis Prediction

Diabetes prediction models help in early detection and prevention. These models use patient health metrics to assess diabetes risk.

Key Concepts:

* Binary classification in healthcare
* Handling medical data privacy
* Model interpretability requirements
* Clinical feature importance

## Objective

To understand how machine learning predicts diabetes risk. To implement classification models for medical diagnosis. To evaluate models using medical metrics.

## Learning Outcomes

By the end of this experiment, participants will be able to:

1. Preprocess medical data with privacy considerations
2. Implement logistic regression for diagnosis
3. Evaluate using sensitivity and specificity
4. Interpret clinical feature importance

Instructions for Conducting the Experiment

1. Preparation:

from sklearn.linear\_model import LogisticRegression from sklearn.metrics import confusion\_matrix

1. Data Preprocessing:

df = pd.read\_csv('diabetes.csv') X = df.drop('Outcome', axis=1) y = df['Outcome']

from sklearn.preprocessing import StandardScaler scaler = StandardScaler()

X = scaler.fit\_transform(X)

1. Model Training:

model = LogisticRegression() model.fit(X\_train, y\_train)

1. Evaluation:

y\_pred = model.predict(X\_test)

tn, fp, fn, tp = confusion\_matrix(y\_test, y\_pred).ravel() print(f"Sensitivity: {tp/(tp+fn)}")

print(f"Specificity: {tn/(tn+fp)}")

# Experiment 4: Understanding Credit Card Default Prediction Models

Introduction to Default Prediction

Credit card default prediction helps financial institutions assess customer risk. These models predict likelihood of payment default.

## Key Concepts:

* Risk assessment modeling
* Customer segmentation
* Financial feature engineering
* Regulatory compliance

## Objective

To understand credit risk prediction models.

To implement classification algorithms for default prediction. To evaluate models using financial metrics.

## Learning Outcomes

By the end of this experiment, participants will be able to:

1. Preprocess financial data
2. Implement gradient boosting for risk prediction
3. Evaluate using precision-recall metrics
4. Interpret risk factors

## Instructions for Conducting the Experiment

1. Preparation:

from xgboost import XGBClassifier

from sklearn.metrics import precision\_recall\_curve

1. Data Preprocessing:

df = pd.read\_csv('credit\_default.csv')

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

from imblearn.over\_sampling import SMOTE smote = SMOTE()

X, y = smote.fit\_resample(X, y)

1. Model Training:

model = XGBClassifier() model.fit(X\_train, y\_train)

1. Evaluation:

y\_pred = model.predict\_proba(X\_test)[:,1]

precision, recall, \_ = precision\_recall\_curve(y\_test, y\_pred) print(f"Average Precision: {np.mean(precision)}")















