**FML (Fundamentals of Machine Learning) Lab**

**AKSHAT MAURYA (241058023)**

Lab 1: Introduction to Python and Basic Syntax  
1. Printing and Comments:   
2. Data Types and Variables:   
3. Basic Operators:   
4. Code Structuring Basics:

**Lab 2:** User Input, Convert User Input, Comparison Operators, Exception Handling, Try-Except Structure, Functions, Loops and Iterations.

**Lab 3:** String Manipulation  
In Lab 3, we dived into Python's string manipulation capabilities  
1. Basic String Operations: We explored string concatenation, repetition, and slicing, allowing us to efficiently combine and retrieve specific portions of text.  
2. String Functions and Methods: We practiced using built-in methods like `.upper()`, `.lower()`, `.find()`, `.replace()`, and `.strip()` for modifying and cleaning strings.

**Lab 4:** Lists, Tuples, and Dictionaries, Time, lambda function.

**Lab 5: Data Pre-processing in Python**

In this lab, we focused on data pre-processing techniques using Python, which are crucial steps before applying machine learning algorithms.

1. **Data Import and Overview**
   * Import necessary libraries and mount Google Drive to access the dataset.
   * Load and display the first few rows of the dataset.
2. **Extract Features and Labels**
   * Separate features (X) from the label (Y), with X being all columns except the last and Y being the last column.
3. **Handling Missing Data**
   * Impute missing values in specified columns (e.g., 'Age' and 'Salary') with the mean value.
4. **Encoding Categorical Data**
   * Label-encode the 'Country' column to convert it to numeric form.
   * Apply one-hot encoding to 'Country' to create dummy variables.
5. **Encoding the Dependent Variable**
   * Label-encode the dependent variable (Y), such as 'Purchased', to convert 'Yes/No' to numeric form.
6. **Splitting the Dataset into Training and Test Sets**
   * Split data into training (80%) and test sets (20%) to evaluate the model’s performance.
7. **Feature Scaling**
   * Standardize feature values to ensure consistent scaling across features in both training and test sets.

**Lab Summary:** These pre-processing steps are essential for cleaning, encoding, and preparing the data for building accurate and efficient machine learning models.

**Lab 6: Classification and regression**

**Part A: Classification using Decision Tree on Social Network Ads dataset**

1. **Data Import**
   * Load the Social Network Ads dataset from Google Drive.
2. **Extract Features and Labels**
   * Separate the dataset into features (X) and label (Y) to prepare for classification.
3. **Splitting the Data**
   * Split data into training and testing sets (same code as in pre-processing).
4. **Feature Scaling**
   * Standardize the feature values for consistency.
5. **Building a Decision Tree Classifier**
   * Train a Decision Tree Classifier using the training data with entropy criterion.
6. **Making Predictions**
   * Predict purchase likelihood on the test set.
7. **Evaluating Model Performance**
   * Use a confusion matrix to evaluate the classifier’s accuracy.
8. **Accuracy Calculation**
   * Calculate and print the model’s accuracy.

**Part B: Simple Linear Regression on Salary Data**

1. **Data Import**
   * Load the Salary Data dataset from Google Drive.
2. **Extract Features and Labels**
   * Separate Years of Experience as feature (X) and Salary as the label (Y).
3. **Splitting the Data**
   * Split the data into training and testing sets (same code as in pre-processing).
4. **Model Training (Linear Regression)**
   * Train a Linear Regression model using the training data.
5. **Prediction**
   * Predict salaries on the test data.
6. **Visualization**
   * Plot training data, test data, and the regression line to visualize the relationship between Years of Experience and Salary.

**Lab 7: Logistic Regression.**

In this lab, we applied Logistic Regression on the Social Network Ads dataset to predict whether a user would purchase a product based on their Age and Estimated Salary. Following data import, we split the data into training and testing sets and standardized the features for consistency. We then trained a Logistic Regression classifier on the training data and used it to make predictions on the test set. To evaluate the model, we generated a confusion matrix, classification report, and calculated the accuracy, achieving a predictive accuracy of 89%. This exercise highlights the effectiveness of Logistic Regression for binary classification tasks, especially when features are well-preprocessed and scaled.

Lab Summary:

we used **Logistic Regression** to classify user purchases based on their **Age** and **Estimated Salary**, achieving an accuracy of **89%**.

**Lab Eaxm: Midterm exam. Oct 8, 2024 - Customer Purchasing Behaviors.csv**

**precision recall f1-score support**

0 0.89 0.96 0.92 68

1 0.89 0.75 0.81 32

**accuracy** 0.89 100

**macro avg** 0.89 0.85 0.87 100

**weighted avg** 0.89 0.89 0.89 100

**Accuracy**: The Logistic Regression model achieved an accuracy of 89%.

**Lab 8: Feature Selection.**

we focused on Feature Selection techniques using the Iris dataset to improve classification models. Here’s a summary of the key steps and results:

**Variance Threshold:** We dropped features with zero variance. This step did not eliminate any features in the Iris dataset since all features had some variance.

**Univariate Feature Selection (Chi-square Test):** We used the Chi-square test to select the top 2 and then 3 best features. The most informative features based on the Chi-square test were the petal length and petal width.

**Model-based Feature Selection (Random Forest):** We used a Random Forest classifier to select important features based on feature importance scores. Features like petal length and petal width were identified as the most important.

**Classification Pipeline:** We combined feature selection with a KNN classifier and Random Forest model.

The process included:

Feature selection using Chi-square (top 2 features).

Further feature selection using a Random Forest model.

Classification using KNN.

**Accuracy:** The classification pipeline achieved a high accuracy of **97.78%** on the test set.

**Feature Importance:** The petal length and petal width had the highest importance scores from the Random Forest model, highlighting their significance in classifying the Iris species.

**Lab 9: Feature Selection**

**Bagging with Decision Tree and Random Forest Classifiers on Social Network Ads dataset**

1. **Decision Tree Classifier (DTC)**
   * Model: Decision Tree with criterion 'entropy' (information gain).
   * Max Depth: Limited to 5 to reduce overfitting.
2. **Random Forest Classifier (RF) Configurations**
   * **RF with 100 Estimators:** Used 100 trees with a max depth of 5 for balanced complexity.
   * **RF with 10 Estimators and Depth 1:** Used only 10 trees, each with depth 1, resulting in a simpler model.
   * **RF with 50 Estimators and Depth 3:** Increased tree count to 50 and depth to 3 for a more refined model.
   * **RF with 75 Estimators and Depth 3:** Increased trees to 75, keeping depth at 3 for enhanced ensemble accuracy.

**Boosting with AdaBoost on Mushroom Dataset**

1. **AdaBoost with Decision Tree as Base Classifier**
   * Base Model: Decision Tree with max depth 1 (as a weak learner).
   * AdaBoost Parameters: 400 estimators and a learning rate of 1.
   * Model Training and Prediction: Trained AdaBoost model to classify mushrooms as edible or poisonous.

**Lab Summary:**

* **Bagging (Random Forest)**: Enhanced model accuracy by combining multiple decision trees trained on random data subsets, yielding higher accuracy than a single decision tree.
* **Boosting (AdaBoost)**: Achieved 100% accuracy by reducing bias and correcting errors from previous learners, showing strong performance, especially with initially misclassified examples.

**Lab 10: K-Nearest Neighbors (k-NN) Classifier**

**In this lab, we implemented the k-Nearest Neighbors (k-NN) algorithm on the Social Network Ads dataset to predict user purchases based on Age and Estimated Salary. We started by importing the data, pre-processing, and splitting it into features and target labels. The initial k-NN model was trained with \( k=5 \), achieving an accuracy of 82%. Through experimentation, we adjusted the value of \( k \) and observed the accuracy improvements, reaching 83% with \( k=7 \). Next, we applied feature scaling, which significantly enhanced model performance, yielding 95% accuracy when \( k=8 \). We also tested the Minkowski distance with \( p=2 \) (equivalent to Euclidean distance), which confirmed the optimal configuration with 95% accuracy. This lab demonstrates the impact of tuning hyperparameters, such as \( k \), and preprocessing steps like feature scaling on the performance of k-NN. Additionally, it highlights that using different data splits (e.g., 80/20 vs. 75/25) can slightly influence results and underscores that k-NN can be an effective classifier when fine-tuned for the dataset and distance metric.**