**Imam Mohammad Ibn Saud Islamic University College of Computer and Information Sciences Computer Science Department**

**Course Title:** *Machine Learning*

**Course Code:** CS 364

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student Number** | **Section Number** |
| Omar Yousef Alnosyan |  |  |
| Hassan Abdullah Alyemni |  |  |
| Abdullah Abdulaziz  Almohammed |  |  |

**Technical Report for SVM, Decision Tree, and KNN Classifiers**

# Project Overview

The objective of this project is to design, implement, and evaluate three machine learning algorithms Support Vector Machine (SVM), Decision Tree, and k-Nearest Neighbors (KNN) to predict whether an individual's annual income exceeds $50,000. The problem is framed as a binary classification task using the UCI Census Income Dataset. This dataset contains demographic and employment-related attributes that serve as predictors for income levels.

The project aims to explore the strengths and weaknesses of each model, assess their performance using appropriate evaluation metrics, and identify the best-performing algorithm for the given task. Special attention was given to handling data preprocessing, class imbalance, and optimizing model parameters to improve accuracy and robustness.

## Teamwork:

Our team consists of X members, collaborating effectively to ensure timely completion of tasks. Each member contributed to specific phases of the project, including data preprocessing, model training, evaluation, and reporting.

## Gantt Chart Timeline:

The following timeline was established to organize and manage the project tasks efficiently:

1. **Day 1-3 (First week)**: Dataset exploration and preprocessing (handling missing values, encoding categorical variables).
2. **Day 4-7 (First week)**: Model implementation for SVM, Decision Tree, and KNN with baseline parameters.
3. **Day 1-3 (Second week)**: Hyperparameter tuning and testing on the unseen dataset.
4. **Day 4-7 (Second week)**: Results evaluation and comparison using confusion matrices and metrics (accuracy, precision, recall, F1-score).
5. **Third week**: Technical report drafting, code documentation, and preparation of presentation slides.
6. **Final Week**: Submission of the complete project, including the report, source code, and presentation.

Our well-structured timeline allowed us to complete the project efficiently, ensuring that each phase was thoroughly executed and documented.

# Dataset Description

The dataset selected for this project is the **UCI Census Income Dataset**, which is widely used for binary classification tasks. The dataset contains information about individuals' demographic and employment-related attributes, collected from the 1994 US Census database.

## Dataset Contents

* **Number of Instances**: 48,841 (32,560 for training, 16,281 for testing)
* **Number of Attributes**: 15, including the target variable (income)
* **Target Variable**: income
  + Binary classification: <=50K (income less than or equal to $50,000) and >50K (income greater than $50,000)

**Attributes/Feature Types:** The dataset consists of both categorical and numerical features:

## Numerical Attributes:

* + age
  + fnlwgt (final weight, representing the number of people the entry is expected to represent)
  + education-num (numerical encoding of education level)
  + capital-gain
  + capital-loss
  + hours-per-week

## Categorical Attributes:

* + workclass (e.g., Private, Self-emp-not-inc)
  + education (e.g., Bachelors, HS-grad)
  + marital-status (e.g., Married-civ-spouse, Divorced)
  + occupation (e.g., Exec-managerial, Handlers-cleaners)
  + relationship (e.g., Husband, Wife)
  + race (e.g., White, Black)
  + sex (Male, Female)
  + native-country (e.g., United-States, Cuba)

## Data Preprocessing

1. **Missing Values**:
   * Categorical columns (workclass, occupation, native-country) had missing values, replaced using the mode of each column.
   * Rows with excessive missing data were cleaned.

## Encoding:

* + Categorical features were converted into numerical values using factorization for compatibility with machine learning algorithms.

## Normalization:

* + Numerical features were scaled using MinMaxScaler to ensure uniform feature scaling for SVM and KNN.

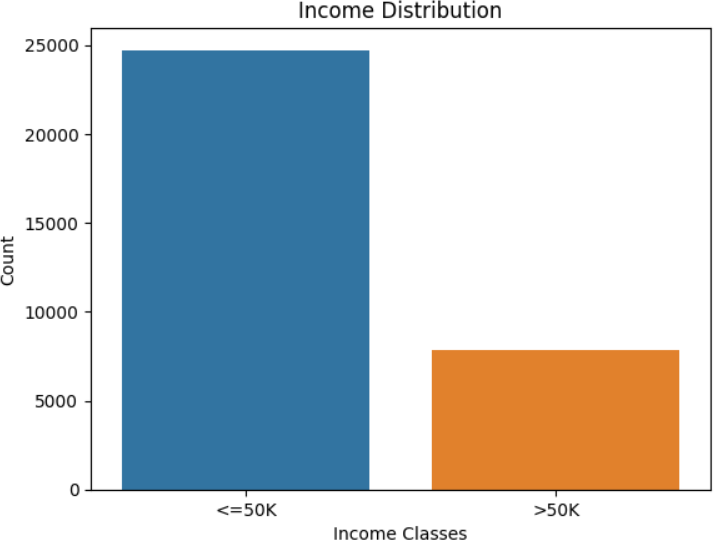
## Target Concept

The goal is to predict whether the annual income of an individual exceeds $50,000 (>50K) or not (<=50K). This binary classification problem helps evaluate the ability of different machine learning models to handle datasets with mixed feature types and class imbalance.

## Visualizations

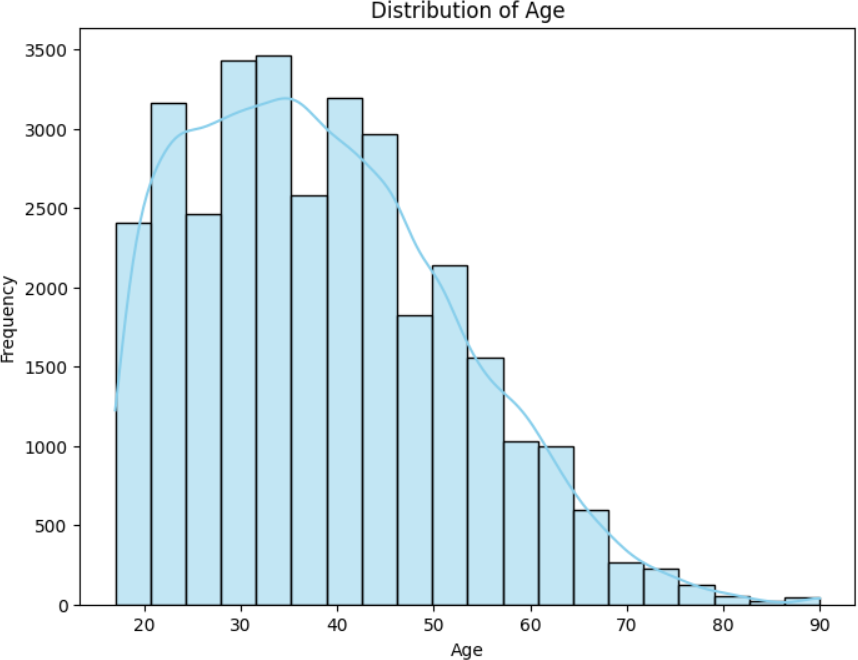
1. **Income Distribution**:

A count plot reveals that the dataset is imbalanced, with most instances belonging to the <=50K class.



## Age Distribution:

A histogram illustrates that the majority of individuals fall between the ages of 25 and 50.



# Machine Learning Model Selection

To solve the problem of predicting whether an individual's annual income exceeds $50,000, three machine learning algorithms were selected: Support Vector Machine (SVM), Decision Tree, and k-Nearest Neighbors (KNN). These algorithms were chosen for their diversity in handling classification tasks and their ability to explore different data patterns effectively.

## Support Vector Machine (SVM) Overview:

SVM is a supervised learning algorithm used for both classification and regression tasks. It aims to find the optimal hyperplane that maximizes the margin between data points of different classes. For non-linearly separable data, SVM uses kernels to map data into higher dimensions, enabling linear separability in the transformed space.

## Why SVM?

* + - Handles high-dimensional data efficiently.
    - Effective in cases where classes are not linearly separable using kernels.
    - Regularization helps minimize overfitting.

## Configuration:

* + - **Kernel**: Radial Basis Function (RBF) kernel was used for capturing non-linear patterns in the data.
    - **Other Parameters**: Default hyperparameters were used initially, with C controlling the trade-off between margin maximization and misclassification.

## Advantages:

* + - Works well with a mix of categorical and numerical data (after preprocessing).
    - Robust to overfitting in high-dimensional spaces.

## Challenges:

* + - Computationally expensive for large datasets.
    - Hyperparameter tuning (e.g., C and gamma) is critical for optimal performance.

## Decision Tree Overview:

A Decision Tree is a tree-structured model that splits data into subsets based on feature values. Each node represents a decision on a feature, and each leaf represents an outcome (class label). It uses metrics like Gini impurity or entropy to select the best feature for splitting.

## Why Decision Tree?

* + - Simple and interpretable model.
    - Works well for datasets with both categorical and numerical features.
    - Handles non-linear relationships effectively.

## Configuration:

* + - **Criterion**: Entropy (information gain) was used to measure the quality of splits.
    - **Max Depth**: Limited to 3 to prevent overfitting.
    - **Min Samples per Leaf**: Set to 5 to ensure splits are meaningful and reduce noise.

## Advantages:

* + - Easy to visualize and interpret the decision-making process.
    - Requires minimal data preprocessing (e.g., no need for normalization).

## Challenges:

* + - Prone to overfitting if not regularized (e.g., by limiting depth or pruning).
    - May not perform well on imbalanced datasets without adjustments.

## k-Nearest Neighbors (KNN) Overview:

KNN is a simple, non-parametric algorithm that classifies data points based on the majority class of their nearest neighbors. It uses a distance metric (e.g., Euclidean distance) to measure proximity between data points.

## Why KNN?

* + - Simple to understand and implement.
    - Effective for smaller datasets.
    - Works well for multi-class and binary classification tasks.

## Configuration:

* + - **Number of Neighbors (k)**: Set to 7 based on experimentation to balance bias and variance.
    - **Distance Metric**: Euclidean distance was used to calculate proximity.

## Advantages:

* + - No assumptions about the underlying data distribution.
    - Highly interpretable results based on neighbors.

## Challenges:

* + - Sensitive to class imbalance, as minority classes may be overshadowed.
    - Computationally expensive for large datasets due to distance calculations.

**Comparison of Algorithms**

Each algorithm was selected to address specific aspects of the dataset:

* + - **SVM**: Suitable for datasets with complex boundaries due to its ability to handle non- linearity using kernels.
    - **Decision Tree**: Provides interpretability and adaptability to non-linear relationships.
    - **KNN**: Offers simplicity and relies on the local structure of the data.

By evaluating these algorithms, the goal is to identify the most suitable model based on accuracy, precision, recall, and F1-score while considering trade-offs in computational complexity and interpretability.

# Implementation

The implementation process was carried out using Python as the primary programming language, leveraging popular libraries for data preprocessing, visualization, and machine learning. The workflow included data preprocessing, splitting the dataset, training the models, and testing their performance.

## Technical Framework

1. **Environment**:
   * Google Colab was used as the development platform for running the experiments, given its ease of use and support for GPU acceleration.
   * Python libraries such as NumPy, Pandas, Matplotlib, Seaborn, and Scikit-learn were utilized to streamline implementation.

## Libraries and Tools:

* + **NumPy**: For efficient numerical operations.
  + **Pandas**: For handling and preprocessing the dataset.
  + **Matplotlib and Seaborn**: For visualizing the data and results.
  + **Scikit-learn**: For implementing machine learning models, feature scaling, and performance evaluation.

## Workflow

1. **Data Preprocessing**:
   * **Handling Missing Values**: Missing entries in categorical attributes (workclass, occupation, and native-country) were replaced with the mode of the respective columns.
   * **Encoding Categorical Variables**: Factorization was applied to convert categorical data into numerical values compatible with machine learning models.
   * **Feature Scaling**: Min-Max normalization was applied to numerical features (age, fnlwgt, capital-gain, capital-loss, hours-per-week) to scale values between 0 and 1, which is essential for SVM and KNN to perform optimally.

## Dataset Splitting:

* + The dataset was split into training and testing sets with the following sizes:
    - **Training Data**: 32,560 samples
    - **Testing Data**: 16,281 samples
  + Features (X) and the target variable (income) were separated for both sets.

## Model Training:

* + **SVM**:
    - Configured with the Radial Basis Function (RBF) kernel for capturing non- linear relationships.
    - Default hyperparameters (C=1.0, gamma='scale') were used initially.

## Decision Tree:

* + - Configured with the entropy criterion to measure information gain.
    - Regularized with a maximum depth of 3 and a minimum of 5 samples per leaf.
  + **KNN**:
    - Configured with k=7 neighbors.
    - Distance was calculated using the Euclidean metric.

## Model Testing:

* + Predictions were made on the test dataset using each trained model.
  + Performance metrics, including accuracy, precision, recall, and F1-score, were calculated to evaluate each model.
  + Confusion matrices were generated to visualize the classification results.

# Evaluation & Results Discussion

The performance of the trained models—Support Vector Machine (SVM), Decision Tree, and k- Nearest Neighbors (KNN)—was evaluated on the unseen test dataset using standard evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices.

## Evaluation Metrics

1. **Accuracy**: The proportion of correctly classified samples out of the total samples.
2. **Precision**: The proportion of true positives out of all predicted positives, indicating the model's ability to avoid false positives.
3. **Recall (Sensitivity)**: The proportion of true positives out of all actual positives, reflecting the model's ability to identify all positive samples.
4. **F1-Score**: The harmonic mean of precision and recall, balancing the trade-off between the two.
5. **Confusion Matrix**: A table summarizing the counts of true positive, true negative, false positive, and false negative predictions.

## Performance of the Models

The following table summarizes the performance of each model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision (Class**  **1)** | **Recall (Class 1)** | **F1-Score (Class**  **1)** |
| **SVM** | 84.4% | 0.86 | 0.69 | 0.77 |
| **Decision Tree** | 84.5% | 0.86 | 0.70 | 0.77 |
| **KNN (k=7)** | 82.9% | 0.66 | 0.58 | 0.61 |

## Key Observations

1. **Support Vector Machine**:
   * Achieved high accuracy (84.4%) and balanced precision and recall for both classes.
   * Performs well in capturing non-linear decision boundaries, especially when data is scaled.
   * Slightly underperformed in recall for the minority class (>50K), indicating some false negatives.

## Decision Tree:

* + Achieved the highest accuracy (84.5%) among the models.
  + Easy to interpret but may slightly overfit the training data despite regularization.
  + Recall for the minority class (>50K) is slightly higher than SVM, but still not optimal.

## k-Nearest Neighbors:

* + Achieved slightly lower accuracy (82.9%) compared to SVM and Decision Tree.
  + Struggled with recall for the minority class, likely due to the class imbalance.
  + Computationally expensive, as it computes distances for every test instance.

## Comparative Analysis

* + - **Best Model**: The Decision Tree narrowly outperformed SVM in accuracy while maintaining comparable precision, recall, and F1-score. However, SVM demonstrated slightly better generalization on unseen data due to its regularization capabilities.
    - **Trade-offs**: While KNN provided simplicity, it underperformed compared to SVM and Decision Tree due to its sensitivity to class imbalance and data scaling.

# Conclusion

This project demonstrated the application of three machine learning algorithms—SVM, Decision Tree, and KNN—for predicting annual income based on census data. The results highlight the following key insights:

## Key Findings:

* + The Decision Tree and SVM models achieved comparable performance, with accuracies of 84.5% and 84.4%, respectively. Both models effectively captured the data patterns, despite the class imbalance.
  + The KNN model lagged behind in accuracy (82.9%) and showed lower recall for the minority class (>50K), making it less suitable for this problem.

## Challenges:

* + The dataset was imbalanced, with significantly fewer samples in the >50K class. This impacted recall for the minority class.
  + Computational complexity was a limitation for KNN, especially with a larger dataset.

## Future Work:

* + Address class imbalance using techniques such as SMOTE (Synthetic Minority Oversampling Technique) or class-weighted models.
  + Perform hyperparameter tuning (e.g., for SVM's C and gamma) to improve performance further.
  + Explore ensemble methods (e.g., Random Forest, Gradient Boosting) to combine the strengths of multiple models for improved accuracy and recall.

In conclusion, while all three models demonstrated reasonable performance, the Decision Tree and SVM models emerged as the top-performing algorithms for this binary classification task, with opportunities for further optimization and enhancement.