HIT391 MACHINE LEARNING: ADVANCEMENTS AND APPLICATIONS

ASSIGNMENT 3

DARWIN GROUP 31

MUSRAT JAHAN S380098
ABHI KUMAR S387006
OWEN TRAN S383410
MEETKUMAR SAVALIYA S382825

Introduction

This report explores implementing and evaluating machine learning techniques on two real-world datasets. The first task covers unsupervised learning using K-Means and Hierarchical Clustering to discover natural user groupings based on interests from the Kaggle Interests Dataset. The goal was to identify patterns without using any predefined labels. The second part involves a classification task using the Titanic: Machine Learning from Disaster competition data. In the second task, three supervised algorithms, including Logistic Regression, Random Forest and Support Vector Machine (SVM) were developed to predict passenger survival based on features like age, gender, and class.

Task 1: Clustering – User Interest Groups

Dataset:

- Rows: 6,340 individuals
- Columns: 217 binary features
  each column represents a hobby or interest
  values are likely 1 (interested) or 0 (not interested)

  Objective: Group individuals into clusters based on similar hobby profiles.

K-means Clustering on Hobby Groups Dataset

Step 1: Data Preprocessing of K-means Clustering

- Dataset Loaded: The dataset contains 6340 rows and 217 feature columns (interests).
- Preprocessing:
  - Removed any non-numeric columns (e.g., group labels).
  - Checked and dropped missing values.
  - Applied **StandardScaler** normalization to scale all features to have mean 0 and unit variance — necessary for K-means.

Step 2: Data Splitting of K-means Clustering

- Split Ratio:
  - 80% → Training
  - 10% → Evaluation
  - 10% → Testing

Used train_test_split to ensure reproducibility with random_state = 42.

Step 3: Model Implementation of K-means Clustering

- Used the KMeans algorithm from the scikit-learn library.
- KMeans partitions the data into a set number of clusters.
- It does this by minimizing the distance between each data point and the nearest cluster center.
- This helps ensure that individuals in the same cluster have similar hobby profiles.
- Set n_clusters = 4, based on prior knowledge that the dataset represents four distinct groups of people.
- KMeans works iteratively, assigning individuals to clusters based on similarity in their hobby responses.
- It then updates cluster centers to better reflect the assigned data points.
- After training, each person is labelled with one of the four clusters.
- These labels can be used to analyse group patterns and shared interests.

Step 4: Model Training of K-means Clustering

- Trained the KMeans model using the full dataset to identify patterns in individuals' hobby preferences.
- The model grouped people into clusters by analyzing similarities in their responses.
- During training, it learned how to assign each person to the most appropriate cluster.
- After training, each person received a cluster label representing their group.
- These cluster labels were used for internal evaluation (to assess clustering quality) and visualization (to observe group patterns).

Step 5: Model Tuning of K-means Clustering

- A manual grid search was used to find the best number of clusters (k) by running KMeans for values from k = 2 to 9.
- For each value of k, the Silhouette Score was computed to evaluate clustering quality.
- The best result was observed at k = 4, confirming the assumption of four natural groups in the dataset.
- This tuning was done on the evaluation dataset to improve overall model performance.

Step 6: Model Evaluation of K-means Clustering

- The final KMeans model was retrained using both the training and evaluation data with the best k = 4.
- It was tested on the testing dataset to check how well it works on new data.
- Silhouette Score was used to measure clustering quality. Example: A score of 0.52 means the clusters are fairly good (moderately separated and consistent).
- If actual group labels were available:
  - Accuracy – how many were grouped correctly.
  - Precision, Recall, F1-score – for measuring detailed performance.
  - Adjusted Rand Index (ARI) – to compare predicted clusters with real groups.

Hierarchical Clustering on Hobby Groups Dataset

Step 1: Data Preprocessing of Hierarchical Clustering

- Loaded the dataset containing hobby and interest responses.
- Removed any existing group labels, since clustering doesn't require them.
- Checked and handled missing values to ensure clean input data.
- Applied StandardScaler to normalize the features so that all hobby responses are on the same scale.

Step 2: Data Splitting of Hierarchical Clustering

- The dataset was divided into three disjoint subsets:
  - Training set – for building the clustering model.
  - Evaluation set – for tuning the model and selecting the best parameters.
  - Testing set – for final evaluation of model performance.

Step 3: Model Implementation of Hierarchical Clustering

- Used Agglomerative Hierarchical Clustering from the scikit-learn library.
- This method builds clusters by repeatedly merging the two closest groups of data points.
- A linkage method (e.g., 'ward', 'average', or 'complete') was selected to determine how distances between clusters are calculated.
- No need to set the number of clusters in advance — it can be decided later by cutting the dendrogram.

- This approach is useful for visualizing the structure of the data and finding natural groupings.

Step 4: Model Training of Hierarchical Clustering

- The Agglomerative Clustering model was trained on the training dataset.
- During training, the model merged similar data points step by step based on distance and the chosen linkage method.
- This process created a hierarchical tree (dendrogram) showing how data points group together.
- The model learned the structure and relationships among individuals based on their hobby responses.
- After training, the dendrogram was analyzed to choose an appropriate number of clusters

Step 5: Model Tuning of Hierarchical Clustering

- Hyperparameter tuning was performed on the evaluation dataset to improve clustering performance.
- The main parameter tuned was the linkage method (ward, average, complete, etc.), which affects how distances between clusters are calculated.
- A manual grid search approach was used to test different linkage options.
- For each option, Silhouette Score was calculated to measure clustering quality.
- The linkage method with the highest silhouette score was selected for the final mode.

Step 6: Model Evaluation of Hierarchical Clustering

- The final Hierarchical Clustering model was trained using both the training and evaluation data.
- It was then tested on the test dataset to evaluate how well it performs on new, unseen data.
- Silhouette Score was used to measure clustering quality. Example: A score of 0.50 would indicate moderately good clustering (some separation and cohesion).
- If actual group labels were available:
  - Accuracy – to see how many individuals were grouped correctly.
  - Precision, Recall, F1-score – to assess performance in detail.
- Adjusted Rand Index (ARI) – to compare predicted clusters with the actual groups

Task 2: Titanic Survival Prediction

Dataset: Originally from Kaggle competition 'Titanic - Machine

Learning from Disaster'. https://www.kaggle.com/competitions/titanic/overview

- train.csv: Contains both features and the target variable (survival) for training the machine learning model. This is the main dataset for the report based on the objectives.
- test.csv: Contains only features, with no survival data (target), to evaluate the model's performance.
- gender_submission.csv: A sample submission file that assumes all female passengers survived.

Variables:

- PassengerId: The Unique identifier assigned to each passenger.
- Survival: Show whether the passenger survived or not (0 = No, 1 = Yes).
- Pclass: Ticket class, representing socio-economic status.
- Name: Full name of the passenger, including title (e.g., Mr, Mrs).
- Sex: Passenger's gender.
- Age: Age of the passenger, in years.
- SibSp: Number of siblings or spouses aboard the Titanic.
- Parch: Number of parents or children aboard the Titanic.
- Ticket: Passenger's ticket number.
- Fare: The fare paid by the passenger for the Titanic journey.
- Cabin: Cabin number where the passenger stayed.
- Embarked: Port of embarkation for the passenger.

Objectives:

The goal is to create a model that can accurately predict whether a passenger survived or not, by using passenger data based on their gender, age, ticket class, family relationships, and port of embarkation, etc.

Step 1: Data Preprocessing

- Data loaded: The dataset contains 891 rows and 12 features.
- Handle Missing Data:

- Fill missing age values with the average or median age.
- Fill missing embarked values with the most common embarkation port.
- Dropped Cabin column due to its excessive missing data.

➢ Create Family Size:
- Created a new feature called FamilySize by adding the number of siblings/spouses (SibSp) and parents/children (Parch).
- This gives a better idea of whether the passenger was alone or traveling with family.

➢ Encode Categorical Variables:

- Convert sex into numbers (male = 0, female = 1).
- Extracted titles (Mr, Mrs, Miss, etc.) from Name and applied One-Hot Encoding to both Title and Embarked columns to turn them into separate ones.

➢ Standardize Numerical Features:

- Scaled Age, Fare, and FamilySize to a standard range (e.g., mean = 0, standard deviation = 1).
- Scaled numerical features using StandardScaler to help neural network training stability.

➢ Drop irrelevant column:
- Dropped irrelevant columns like Name, Ticket, and PassengerId.

Step 2: Data Splitting

➢ The dataset was divided into two disjoint subsets:
- Training Set + (80%): Used to train the model.
- Evaluation Set (20%): Used to tuning and checking performance before testing.

➢ All splits are done using train_test_split and random_state = 42 to keep results reproducible for the model.

Step 3: Model Implementation

▪ Logistic Regression:
- ➢ Used LogisticRegression from scikit-learn.
- ➢ It is a linear classification algorithm that models the probability of a binary outcome based on a logistic function.
- ➢ Trained on the training set and evaluated on the evaluation set.

▪ Support Vector Machine (SVM):
- ➢ Used GridSearchCV for parameter tuning.

- Searched across multiple values of C (like 0.1, 1, 10) and two kernel types – linear and rbf.
- Used 5-fold cross-validation and evaluated using F1-score with concern about balance between precision and recall.

- Random Forest:
  - Used RandomForestClassifier from scikit-learn with default settings.
  - It is an ensemble model that constructs multiple decision trees and combines their outputs through majority voting.
  - It handles both categorical and numerical features well and is effective at reducing overfitting by averaging across trees.
  - Trained on the training set and evaluated on the evaluation set.

Step 4: Model Training

- Logistic Regression:
  - Trained on the training dataset.
  - Used it to predict on the evaluation dataset.
  - Predicted probabilities and decided in whether someone survived or not with binary (0 or 1) based on logistic function output.

- Support Vector Machine (SVM):
  - Trained using the best parameters selected by GridSearchCV.
  - The best model was used to predict and evaluate performance on the evaluation set.

- Random Forest:
  - Trained on the training dataset.
  - Each decision tree in the forest made a prediction on the evaluation set, and the final output was determined by the majority vote from all trees.

Step 5: Model Tuning

- Logistic Regression and Random Forest tuning:
  - No hyperparameter tuning was applied in this task.
  - The model was used with its default configuration and parameters.

- SVM tuning:
  - Tried out multiple values for C and different kernel options.
  - Best result was found with: C = 10, gamma = scale, kernel = linear.
  - This tuned model was used for final evaluation.

- Random Forest tuning:
  - No hyperparameter tuning was applied.
  - Was trained using default parameter.

Step 6: Model Evaluation

- The provided test.csv does not include survival labels and was therefore not used for evaluation. Instead, the evaluation subset was used.
- Evaluated the models using standard metrics like:
- Accuracy: Measures the overall proportion of correctly predicted instances.
- Precision: Indicates how many of the predicted positive cases were correctly positive.
- Recall: Measures how many of the actual positive cases were correctly predicted (True Positive Rate)
- F1 Score: The harmonic mean of precision and recall, especially useful when dealing with imbalanced classes.
- Tuned SVM outperformed the other two algorithms' models.
- Logistic Regression provided a baseline for comparison with the other models.

Results on Evaluation Set

Metrics	Logistic Regression	Tuned SVM (Best Model)	Random Forest
Accuracy	0.7933	0.8883	0.8268
Precision	0.7534	0.8750	0.7867
Recall	0.7432	0.8514	0.7973
F1 Score	0.7483	0.8630	0.7919

Conclusion:

In conclusion, both tasks illustrated machine learning methods for identifying trends and formulating predictions in various settings. The clustering algorithms showed user behavior. On the other hand, the classification methods demonstrated how well machine learning works to solve practical issues by offering a firm grasp of survival prediction using the Titanic dataset.

Contribution:

Name	**Task**
Musrat Jahan	Report: Task 1 & Task 2
Meet Kumar	Task 1 Coding
Owen Tran	Task 2 Coding
Abhi Kumar	Report : Task 2 (SVM)