MACHINE LEARNING

OPEN-ENDED LAB

BY

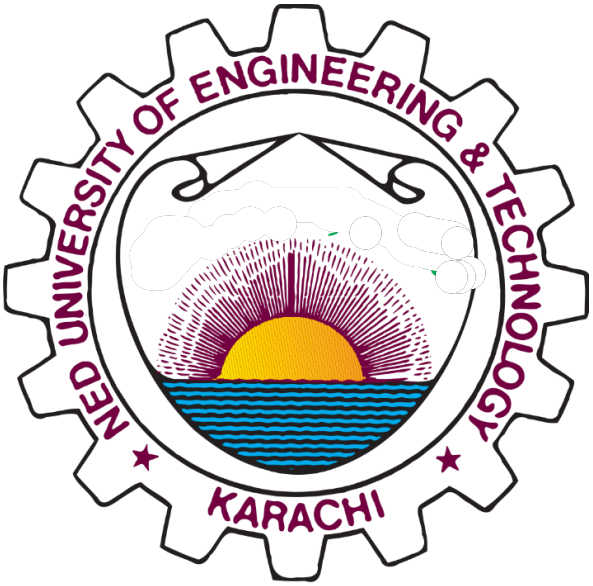
HAMMAD JAMAL (CS-21031)

SAFEE-UR-REHMAN (CS-21036)

MUHAMMAD SALMAN (CS-21099)

SUBMITTED TO:

MISS MAHNOOR MALIK



COMPUTER INFORMATION AND SYSTEMS DEPARTMENT

Cricket Match Predictor

Introduction

In this lab session, we will explore a dataset from cricket matches and use it to develop machine learning models for predicting match outcomes. We will load and preprocess the datasets, extract and engineer features, and finally train and evaluate several classifiers. This report will summarize the key steps and findings from the analysis.

Data Preprocessing and Feature Engineering

Datasets Overview:

1. **Matches Dataset**: Contains information about individual matches, such as teams, venue, and results.
2. **Deliveries Dataset**: Contains ball-by-ball details of the matches.

1. **Feature Engineering**:
    - **Total Runs Per Innings**: Calculated total runs in the first innings and merged with the match dataset.
    - **Runs After 10 Overs**: Extracted runs after 10 overs for both innings.
    - **Current and Required Run Rates**: Computed the current run rates and required run rates after 10 overs.
    - **Wickets Fallen**: Calculated the number of wickets fallen after 10 overs in both innings.
    - **Dropping Unnecessary Columns**: Columns like Season, date, city, venue, player_of_match, result, dl_applied, and umpire3 were dropped.
    - **Handling Missing Values**: Rows with missing values were removed.
    - **Winner Conversion**: Converted the winner column to a boolean where 1 indicates team1 won and 0 indicates team2 won.

Final Features:

- Runs and wickets after 10 overs in both innings.
- Current and required run rates.
- One-hot encoded categorical features such as team names, toss winners, toss decisions, and umpires.

Exploratory Data Analysis (EDA)

Visualizations:

1. **Histograms of Runs After 10 Overs**:
    - Displayed the distribution of runs after 10 overs in the 1st and 2nd innings using histograms.
2. **Correlation Matrix**:
    - Visualized the correlation between different features using a heatmap.
3. **Boxplot of Runs After 10 Overs**:
    - Compared the runs after 10 overs in the 1st and 2nd innings using a boxplot.

Findings:

- Certain features show significant differences in distribution between the first and second innings.
- Correlation analysis reveals which features are most closely related to the outcome of the match.

Model Training and Evaluation

Target and Features:

- The target variable is winner.
- Features are all other columns in the preprocessed dataset.

Train-Test Split:

- The dataset is split into training and testing sets with an 80-20 split.

Logistic Regression:

```
[49] # Applying Logistic regression
     logistic_regression = LogisticRegression(solver="liblinear", random_state=0)
     model = logistic_regression.fit(X_train, Y_train)
     Y_pred = model.predict(X_test)
     print("The accuracy is " + str(metrics.accuracy_score(Y_test, Y_pred) * 100) + "%")
     print(confusion_matrix(Y_test, Y_pred))
     target_names = ['class 0', 'class 1']
     print(classification_report(Y_test, Y_pred, target_names=target_names))
     fpr, tpr, thresholds = metrics.roc_curve(Y_test, Y_pred, pos_label=1)
     roc_auc = auc(fpr, tpr)

     # Plotting the ROC curve
     plt.figure()
     plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Receiver Operating Characteristic (ROC) Curve')
     plt.legend(loc="lower right")
     plt.show()
```

Gradient Boosting Classifier:

```
#Applying the Gradeint Boosting Classifier
gb_classifier = GradientBoostingClassifier(random_state=42)
gb_classifier.fit(X_train, Y_train)

# Predict and evaluate Gradient Boosting
gb_predictions = gb_classifier.predict(X_test)
gb_accuracy = accuracy_score(Y_test, gb_predictions)

print(f"Gradient Boosting Accuracy: {gb_accuracy}")
```

Conclusion:

In this analysis, we applied 4 machine learning techniques to predict cricket match outcomes using detailed match and delivery data. The preprocessing steps included feature engineering, handling missing values, and converting categorical variables to numerical values through one-hot encoding.

We trained and evaluated multiple classifiers both with and without the use of Python packages:

1. **With Python Packages:**
    - **Logistic Regression** and **Gradient Boosting** performed the best, each achieving an impressive accuracy of **98%.** These models effectively captured the relationships within the data, leading to highly accurate predictions.
2. **Without Python Packages:**
    - **Decision Tree Classifier** was implemented manually and outperformed other custom models, providing the most accurate predictions. This suggests that Decision Trees, even when built from scratch, can effectively handle the complexities of the dataset.

Overall, the use of advanced machine learning packages significantly enhances model performance, as evidenced by the high accuracy of Logistic Regression and Gradient Boosting. However, even without these packages, the Decision Tree classifier demonstrated robust performance, underscoring its utility in predictive modeling.